Titre: Title:	Efficient Learning of Markov Blanket and Markov Blanket Classifier
Auteur: Author:	Shunkai Fu
Date:	2010
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Fu, S. (2010). Efficient Learning of Markov Blanket and Markov Blanket Classifier [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/400/

UNIVERSITÉ DE MONTRÉAL

EFFICIENT LEARNING OF MARKOV BLANKET AND MARKOV BLANKET CLASSIFIER

SHUNKAI FU

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÉSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR

(GÉNIE INFORMATIQUE)

AOÛT 2010

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

EFFICIENT LEARNING OF MARKOV BLANKET AND MARKOV BLANKET CLASSIFIER

présentée par : FU Shunkai

en vue de l'obtention du diplôme de : Philosophiae Doctor

a été dûment accepté par le jury d'examen constitué de :

M. GAGNON Michel, Ph.D, président

M. DESMARAIS Michel ,Ph.D, membre et directeur de recherche

M. PAL Christopher J., Ph.D, membre

Mme. PRECUP Doina, Ph.D, membre

DEDICATION

The author wishes to dedicate this dissertation to my family, who offered me unconditional love and support throughout the course of this thesis.

ACKNOWLEDGEMENTS

My utmost gratitude goes to my master and doctoral program supervisor, Dr. Michel C. Desmarais for bringing me to Ecole Polytechnique de Montreal, for his expertise, kindness, and most of all, for his patience. I believe that one of the main gains of this 5-year program was working with Dr. Michel C. Desmarais and gaining his trust and friendship. My thanks and appreciation goes to the thesis committee members, Dr. Michel Gagnon, Dr. Christopher Pal and Dr. Doina Precup. I do owe thanks to Dr. Wai-Tung Ho and Dr. Spisis Demire for their sharing and guide during my time in SPSS. Also, I thank my wife, Shan Huang who stands beside me and encourages me constantly. My thanks goes to my children, JingYi and JingYao for giving me happiness and joy. Finally, I would like to thank my parents whose love is boundless.

CONDENSÉ EN FRANÇAIS

Cette thèse porte sur deux sujets très reliés, à savoir la classification et la sélection de variables. La première contribution de la thèse consiste à développer un algorithme pour l'identification du sous-ensemble optimal de variables pour la classification d'une variable cible dans un cadre bayésien, sous-ensemble que l'on désigne par la Couverture de Markov d'une variable cible. L'algorithme développé, IPC-MB, affiche une performance prédictive et une complexité calculatoire se situant au niveau des meilleurs algorithmes existants. Cependant, il est toutefois le seul algorithme affichant les meilleures performances sur les deux plans simultanément, ce qui constitue une percée importante au plan pratique.

La Couverture de Markov d'une variable cible est un sous ensemble qui ne comporte pas de structure, notamment le réseau bayésien des variables impliquées. La seconde contribution consiste à exploiter les résultats intermédiaires de l'algorithme IPC-MB pour améliorer l'induction de la structure du réseau bayésien correspondant à la Couverture de Markov d'une variable cible. Nous démontrons empiriquement que l'algorithme pour induire la structure du réseau bayésien est légèrement plus efficace qu'un algorithme standard comme PC qui n'utilise pas les données intermédiaires de IPC-MB.

La sélection des variables pertinentes pour une tâche de classification est un problème fondamental en apprentissage machine. Il consiste à réduire la dimensionalité de l'espace des solutions en éliminant les attributs qui ne sont pas pertinents, ou qui le sont peu. Pour la tâche classification, un jalon important vers la résolution de ce problème a été atteint par Koller et Sahami [1]. Basé sur les travaux de Pearl dans les réseaux bayésiens [2], il a établi que la Couverture de Markov (*Markov Blanket*) d'une variable T représentait le sous-ensemble optimal d'attributs pour la prédiction de sa classe. Nous le dénotons $\mathbf{MB}_T$ lorsque la variable cible est connue et par **MB** autrement.

Induire $\mathbf{MB}_T$ étant donné un réseau bayésien est un problème trivial. Cependant, l'apprentissage de la structure d'un réseau bayésien à partir de données est un problème reconnu comme NP difficile [3]. Pour un grand nombre de variables, l'apprentissage d'un réseau bayésien est en pratique très difficile non seulement à cause de la complexité calculatoire, mais aussi à cause de la quantité de données requises pour des problèmes dont la dimensionalité est très grande. Souvent, le problème de la dimensionalité est contourné en imposant des contraintes sur la

structure comme c'est le cas avec les réseaux bayésiens naïfs [4, 5], qui sont probablement les plus répandus. Leur complexité calculatoire est relativement faible, n'ayant pas à effectuer un apprentissage de la structure du réseau, et leur efficacité est très souvent relativement bonne malgré les hypothèses fortes qu'ils imposent.

Une des extensions des réseaux bayésiens naïfs est le formalisme de réseaux bayésiens naïf arborescent (Tree-Augmented Naïve Bayes, ou TAN) [6]. Les TAN sont généralement plus performants que les réseaux bayésiens naïfs en permettant certaines formes de dépendance parmi les attributs. Cependant, ils repondent néanmoins sur des hypothèses fortes qui peuvent les rendre invalides en général. Du fait que les réseaux bayésiens ne font pas d'hypothèses fortes sur les données, on s'attend que leur performance pour la classification soit meilleure que pour un réseau bayésien naïf ou un TAN [7]. Cependant, il faut noter que pour la tâche de classification, seulement un sous-ensemble du réseau bayésien est effectif pour la prédiction, c'est-à-dire la Couverture de Markov du nœud cible, $\mathbf{MB}_T$. Lorsque ce sous-ensemble $\mathbf{MB}_T \cup \{T\}$ est utilisé aux fins de classification, nous y référerons par $\mathbf{MBC}_T$, c'est-à-dire le classificateur basé sur la Couverture de Markov. En général, le classificateur $\mathbf{MBC}_T$ est considérablement plus petit que le réseau bayésien et sa performance est en théorie équivalente à celle du réseau bayésien complet. L'induction de **MB** et **MBC** sont deux problèmes très près l'un de l'autre, bien que l'induction de **MB** peut s'avérer être une étape indépendante.

Cette thèse aborde le problème de l'efficacité de l'apprentissage de **MB** et **MBC** à partir d'un échantillon de données limité. L'objectif premier est de fournir un algorithme général de sélection des variables, ou attributs, tel que requis pour différentes tâches de classification, ou même de forage de données. Notre première contribution est de définir un algorithme qui élimine les attributs non pertinents d'un **MB** (ou **MBC**) sous *l'hypothèse de la fidélité* (voir plus loin), lorsque la Couverture de Markov d'une variable cible T est unique et composée des parents de T ($\mathbf{Pa}_T$), de ses enfants ($\mathbf{Ch}_T$) et de ses conjoints, ou *spouses,* ($\mathbf{Sp}_T$) [2]. Selon notre revue de la littérature, il existe au moins neuf travaux publiés depuis 1996 portant sur l'apprentissage de la couverture de Markov, c'est-à-dire depuis que le concept a été démontré être le sous-ensemble optimal d'attributs pour la prédiction et malgré le fait qu'il est connu depuis 1988 [1][2].

Tous les algorithmes connus peuvent être regroupés en deux catégories : (1) ceux qui dépendent de la propriété d'indépendance conditionnelle $I(T, X|\mathbf{MB}_T)$, où T est considéré indépendant de

toutes les autres variables étant donné les valeurs connues de $\mathbf{MB}_T$; et (2) les algorithmes qui reposent sur l'information topologique, c'est-à-dire la recherche des parents, enfants et conjoints du nœud cible. IAMB [8] est l'exemple le plus représentatif des algorithmes du premier groupe. Sa complexité calculatoire et son implémentation sont toutes deux d'une grande simplicité. IAMB comporte deux phases: la phase de croissance et celle de décroissance. Chaque phase nécessite la vérification de savoir si une variable X est indépendante de T étant donné un ensemble de nœuds candidats de la Couverutre de Markov, $\mathbf{MB}_T^C$, puis d'enlever ou d'ajouter des nœuds de cet ensemble de candidats. PCMB [9] est la contribution la plus récente aux algorithmes avant nos travaux et il est un exemple du second groupe. Ce fut en réalité le premier et alors le seul algorithme dont la preuve a été faite qu'il peut induire la Couverture de Markov, bien que ce n'est toutefois pas le seul qui s'est appuyé sur l'information topologique pour le faire.

Malgré ces avancements, la recherche d'un algorithme qui peut à la fois garantir de recouvrer la Couverture de Markov et le faire en un temps raisonnable et avec un ensemble de données réaliste demeure un objectif non atteint. Par exemple, KS [1] est un algorithme approximatif (il ne peut garantir de recouvrer la Couverture de Markov); IAMB est un algorithme simple qui peut fournir cette garantie, mais il impose une quantité extraordinaire de données afin d'arriver à un résultat acceptable pour des problèmes pratiques; MMPC/MB [10] et HITON-PC/MB [11] représentent les premiers essais pour améliorer l'efficacité en regard des données par l'exploitation de données topologiques, mais il a été démontré qu'ils n'offrent pas la garantie de recouvrer la Couverture de Markov [9]; PCMB a suivi la découverte de MMPC/MB et de HITON-PC/MB, et ils peuvent effectivement fournir de bien meilleurs résultats que IAMB pour les mêmes données. Cependant, PCMB est beaucoup plus lent que IAMB, et nos résultats suggèrent même qu'il peut nécessiter plus de temps que l'algorithme PC (voir Chapitre 4). Nous proposons l'algorithme IPC-MB [12-14] afin d'offrir une solution qui vise à la fois à fournir des résultats en un temps rapide et avec une quantité de données réaliste. Cet algorithme est de la seconde catégorie, c'est-à-dire qu'il utilise l'information topologique pour dériver la couverture de Markov.

Tout comme les algorithmes MMPC/MB [10], HITON-PC/MB [11] et PCMB, l'algorithme IPC-MB divise l'apprentissage de $\mathbf{MB}_T$ en deux phases séparées, l'induction de $\mathbf{PC}_T$ et de $\mathbf{Sp}_T$. Dans la première phase, IPC-MB effectue une recherche pour trouver les voisins immédiats du nœud et elle est commune aux algorithmes PCMB, MMPC/MB et HITON-PC/MB. Cependant, alors que

ces algorithmes effectuent une série de tests afin de déterminer si un nœud X n'est PAS indépendant du nœud cible T étant donné tous les ensembles possibles de conditions, c'est-à-dire $\sim I(T, X|\mathbf{Z}_T)$ où $\mathbf{Z}_T \subseteq \mathbf{U}\backslash\{T\}$, IPC-MB présume initialement que toutes les variables du domaine à l'exclusion de T (c.-à-d. $\mathbf{U}\backslash\{T\}$) sont des candidats à $\mathbf{PC}_T$. Puis, l'algorithme élimine les variables une à une si X est indépendant de T étant donné un ensemble de conditions $\mathbf{Z}_T$ quelconque. Parce que la majorité des réseaux ne sont pas denses en pratique et que IPC-MB commence par des ensembles conditionnels vides pour les élargir un nœud à la fois, il lui est possible d'éliminer la majorité des faux candidats avec un petit ensemble de conditionnels, ce qui entraîne un gain en termes de calculs et de données nécessaires. Bien que certains descendants de T peuvent demeurer dans $\mathbf{PC}_T$, ils sont rapidement éliminés en réexecutant la même recherche pour chaque $X \in \mathbf{PC}_T^C$ (candidats de $\mathbf{PC}_T$ qui est le résultat de la recherche précédente) afin de déterminer si $T \in \mathbf{PC}_X^C$. De plus, en reconnaissant que tous les conjoints sont contenus dans l'union des résultats des recherches pour $\mathbf{PC}_T$, c.-à-d. $\bigcup_{X\in\mathbf{PC}_T} \mathbf{PC}_X^C$, et que seulement les véritables conjoints contenus dans $\mathbf{PC}_X^C$ seront dépendant de T conditionnellement à l'ensemble séparateur trouvé précédemment plus X, une quantité importante de ressources est économisée en comparaison avec PCMB afin de dériver $\mathbf{Sp}_T$.

Nous faisons la preuve que l'algorithme IPC-MB est valide et comparons sa performance avec les algorithmes qui sont actuellement l'état de l'art, notamment [10], PCMB [9] et PC [15]. Les expériences effectuées avec des échantillons de données générées à partir de réseaux bayésiens connus, notamment des réseaux de petites tailles comme Asia qui compte huit nœuds, des réseaux moyenne envergure comme Alarm et PolyAlarm (une version polyarborescence, *polytee*, de Alarm) avec 37 nœuds, et des réseaux plus grands comme Hailfinder (56 nœuds) et Test152 (152 nœuds). Nous mesurons la performance des algorithmes en termes de précision, rappel et de distance ($\sqrt{\text{precision}^2 + \text{distance}^2}$). Le temps de calcul est mesuré en termes de nombre de tests d'indépendance conditionnelle (CI) et de nombre de *passes* qui doivent être effectuées sur les données (relectures des données), car une seule passe n'est généralement pas suffisante pour mettre en cache toutes les fréquences requises en mémoire. Ces mesures sont couramment utilisées, car elles sont indépendantes du matériel utilisé et représentent la grande partie des ressources calculatoires consommées pour ce type d'algorithmes.

Les résultats démontrent que, IPC-MB fournit (1) un niveau de performance nettement supérieur à IAMB pour une quantité d'observations équivalente, atteignant jusqu'à 80% en réduction de distance (mesurée par rapport au résultat idéal), (2) a une performance légèrement supérieure à PCMB et PC (toujours à quantité de données égales), (3) nécessite jusqu'à 98% moins de tests CI que PC et 95% moins que PCMB, et (4) en moyenne les tests CI comportent un ensemble conditionnel relativement plus petit par rapport à IAMB et PCMB (ce qui est en bonne partie à la source des améliorations observées). Nous pouvons donc conclure que les stratégies d'apprentissage de $\mathbf{PC}_T$ et $\mathbf{Sp}_T$ adoptées pour IPC-MB sont très efficaces et permettent un gain significatif pour atteindre l'objectif d'induire la Couverture de Markov avec un rapport réaliste de temps et de données.

Étant donné le résultat de IPC-MB, c.-à-d. $\mathbf{MB}_T$, les algorithmes conventionnels pour induire la structure d'un réseau bayésien peuvent être appliqués pour recouvrir **MBC** autre modification puisque l'apprentissage de $\mathbf{MB}_T$ est indépendant d'eux. La complexité de l'apprentissage de la structure devrait être considérablement réduite en comparaison de l'apprentissage induit de l'ensemble des variables du domaine **U**. Nous avons réalisé une autre étude dans le cadre de la thèse en appliquant l'algorithme PC pour l'apprentissage de la structure étant donné $\mathbf{MB}_T$, l'algorithme IPC-MB+PC, et avons observé un temps de calcul considérablement réduit. En fait, le résultat de IPC-MB peut être considéré comme la sélection de variables d'un problème et être utilisé dans un grand nombre d'algorithmes de prédiction. L'algorithme a d'ailleurs été développé par l'auteur lorsqu'à l'emploi de SPSS en 2007 et il est actuellement intégré au module Clémentine 12 pour dériver un **MBC**.

Une seconde contribution de cette thèse est l'extension de IPC-MB pour induire la structure d'un **MBC** directement sans avoir à dériver l'ensemble du réseau bayésien au préalable comme la solution IPC-MB+PC le fait, ce qui constitue une première à notre connaissance. Cet algorithme est nommé IPC-MBC (ou IPC-BNC dans une publication antérieure) [16]. Tout comme IPC-MB, il repose sur une recherche locale afin de déterminer les voisins d'une variable.

Étant donné une variable cible $T \in \mathbf{U}$ et les données D, l'algorithme IPC-MBC peut être divisé en 5 étapes, après une initialisation où le nœud T est assigné à une liste de nœuds « visités », *Scanned*=$\{T\}$.

1. **Induction des liens entre T et $\mathbf{PC}_T^C$.** IPC − MBC commence avec un graphe initial $\mathbb{G}$ dans lequel T est connecté avec tous les nœuds autres nœuds de $\mathbf{U}$, sans toutefois spécifier de direction aux liens. Puis, les nœuds dont le test CI indique une indépendance sont alors considérés non connectés. Les tests de CI commencent avec un ensemble conditionnel vide puis incrémentent cet ensemble d<un nœud à la fois jusqu'à ce que tous les tests possibles soient effectués. À la fin du processus, l'ensemble des nœuds connectés à T qui reste, $\mathbf{PC}_T^C$, contient tous les liens entre T et $\mathbf{PC}_T$, les parents et enfants réels de T, mais il contient aussi des faux positifs.

2. **Élimination des faux positifs de $\mathbf{PC}_T^C$, ajout des liens entre tous les nœuds de $\mathbf{PC}_T^C$ et recueil des conjoints candidats.** La seconde étape consiste à établir un lien non-dirigé entre tous les nœuds de $\mathbf{PC}_T^C$ aux autres nœuds de $Y \in \mathbf{U} \backslash Scanned$, pour obtenir $\mathbf{PC}_X^C = \{Y | (Y - X) \in \mathbb{G}\}$ (c.-à-d. tous les Y connectés à un nœud quelconque Z dans $\mathbb{G}$) . Puis la procédure appliquée à l'étape 1 est répétée pour tout $X \in \mathbf{PC}_X^C$ afin d'éliminer les faux liens de dépendance, après quoi chaque X est ajouté à la liste *Scanned*. À cette étape, l'ensemble des liens non-dirigés et des nœuds restants forment un graphe $\mathbb{G}$ contenant (1) uniquement les véritables parents et enfants de T (c.-à-d. $\mathbf{PC}_T$, en présumant des tests CI fidèles) et (2) les liens entre ces parents. Les liens adjacents à $X \in \mathbf{PC}_T$ sont donc des candidats conjoints, $\mathbf{Sp}_T^C$.

3. **Identification des véritables conjoints, $\mathbf{Sp}_T$, ajout des liens entre conjoints eux-mêmes et entre les conjoints et les véritables parents de T, $\mathbf{PC}_T$.** Pour chaque $X \in \mathbf{PC}_T$, on identifie $\mathbf{PC}_X^C$, où $\mathbf{PC}_X^C = \{Y | (X - Y) \in \mathbb{G}\}$ et où $\mathbf{Sp}_T^C = \cup_X \mathbf{PC}_X^C$. Puis, pour chaque $\forall Y \in \mathbf{PC}_X^C \backslash Scanned$, si Y est dependent de T conditionnellement à $Sepset_{T,Y} \cup \{X\}$, alors Y est un veritable conjoint de T, et nous obtenons une structure en V : $Y \rightarrow X \leftarrow T$. De plus, pour ce Y, nous ajoutons des liens non orientés avec chaque $Z \in \mathbf{U} \backslash Scanned$ dans $\mathbb{G}$. Finalement, la procédure similaire permettant de déterminer les faux positifs de $\mathbf{PC}_Y^C$ tel qu'appliquée précédemment aux liens entre Y et $Z \in \mathbf{Sp}_T \cup \mathbf{PC}_T$ qui restent dans $\mathbb{G}$. Comme chaque véritable conjoint de Y est traité de la même façon, tous les liens entre les conjoints, $\mathbf{Sp}_T$, seront identifiés, de même que ceux entre $\mathbf{Sp}_T$ and $\mathbf{PC}_T$.

4. **Élimination des nœuds n'appartenant pas à $\mathbf{PC}_T$ et $\mathbf{Sp}_T$.** L'étape précédente ajoute des nœuds n'appartenant pas à $\mathbf{PC}_T$ **et** $\mathbf{Sp}_T$ à travers le calcul de $\mathbf{Sp}_T^C = \cup_X \mathbf{PC}_X^C$. Ces nœuds sont

éliminés par une procédure similaire à celle de l'étape 2. Le graphe résultant $\mathbb{G}$ comprte alors une structure proche de celle de $\mathbb{MBC}_T$ contenant certains liens dirigés obtenus à travers la structure en V, et la majorité non dirigés.

5. **Orientation des liens.** Une procédure relativement standard est appliquée à $\mathbb{G}$ obtenu de l'étape précédente pour orienter tous les liens et obtenir la structure finale de $\mathbb{MBC}_T$.

L'algorithme IPC-MBC est prouvé correct. Lors de nos tests empiriques, nous avons comparé sa performance de classification (précision, rappel et distance) et son efficience en termes de nombre de tests CI et de passes de données avec celles de PC et IPC-MB+PC (c.-à-d. l'apprentissage de la structure avec l'algorithme PC appliqué sur le produit de IPC-MB). Les mêmes données que celles utilisées pour l'étude de IPC-MB ont été utilisées. Sans surprise, IPC-MBC et IPC-MB+PC sont tous deux plus efficaces que PC, avec un gain de l'ordre de 95%, sans perte au plan de la performance. D'autre part, IPC-MBC affiche un léger gain de performance par rapport à IPC-MB+PC. Quant à son efficacité, on ne peut garantir que IPC-MBC nécessitera moins de tests CI que IPC-MB+PC, mais il nécessite moins de passes sur les données. Ces différences peuvent s'expliquer du fait que IPC-MB et PC n'échangent aucune information intermédiaire alors que IPC-MBC réutilise les mêmes tests CI à la fois pour l'induction de la structure comme pour la sélection des nœuds, ce qui lui confère une meilleure efficacité lors d'une même passe sur les données et influence sa performance.

Outre les deux contributions principales présentées, nous discutons de la question de fiabilité des tests CI et de son influence sur le résultat des algorithmes, ainsi que des actions à prendre advenant le cas de tests non fiables. Une piste derecherche intéressante serait d'explorer le comportement de IPC-MB sous un modeinspiré de la notion d'Oracle en tests logiciels [4]. Le principe consiste à substituer la valeur du test d'indépendance par le résultat "vrai", c'est-à-dire le résultat conforme au réseau Bayésien qui a servi à générer les données aléatoires. Dans un tel mode, deux hypothèses importantes sont alors forcées d'être respectées : (1) celui de la fidélité des données au réseau sous-jacent et (2) la fiabilité du test conditionnel est alors assurée. Une comparaison de la performance du mode " Oracle" avec celle du mode de simulation original permettrait ainsi d'explorer l'impact du non-respect des hypothèses sous-jacentes à IPC-MB.

De plus, pour aborder la question de l'efficacité qui demeure un problème pour des applications réelles, nous présentons uneesquisse d'un algorithme pour paralléliser IPC-MB et un autre d'une

heuristique basée sur IPC-MB qui sont tous deux susceptibles d'améliorer la valeur pratique de ce type d'algorithmes. Finalement, nous abordons la question d'appliquer des algorithmes pour la recherche d'une structure basée sur le score plutôt que sur des tests CI. Le score correspond ici à la probabilité d'observer la distribution donnée étant donnée un réseau bayésien. Quoique considéré comme une approche prometteuse, leur coût calculatoire était jusqu'ici l'obstacle majeur qui a brimé la recherche de telles solutions. En effet, le nombre de topologies possibles de réseau bayésien croît de façon très rapide en fonction du nombre de variables et devient rapidement impossible à traiter après quelques dizaines de variables et même moins. Mais en considérant que IPC-MB réduit considérablement la dimensionalité de l'espace problème et qu'il nous permet de fixer certains liens entre , et , alors les algorithmes basés sur le score peuvent effectuer un gain d'efficacité important en les combinant avec IPC-MB.

RÉSUMÉ

La sélection de variables est un problème de première importance dans le domaine de l'apprentissage machine et le forage de données. Pour une tâche de classification, un jalon important du développement de stratégies sélection de variables a été atteint par Koller et Shamai [1]. Sur la base des travaux de Pearl dans le domaine des réseaux bayésiens (RB) [2], ils ont démontré que la couverture de Markov (CM) d'une variable nominale représente le sous-ensemble optimal pour prédire sa valeur (classe).

Différents algorithmes ont été développés pour d'induire la CM d'une variable cible à partir de données, sans pour autant nécessiter l'induction du RB qui inclue toutes les variables potentielles depuis 1996, mais ils affichent tous des problèmes de performance, soit au plan de la complexité calculatoire, soit au plan de la reconnaissance.

La première contribution de cette thèse est le développement d'un nouvel algorithme pour cette tâche. L'algorithme IPC-MB [9-11] permet d'induire la CM d'une variable avec une performance qui combine les meilleures performances en terme de complexité calculatoire et de reconnaissance. IPC-MB effectue une recherche itérative des parents et enfants du noeud cible en minimisant le nombre de variables conditionnnelles des tests d'indépendance. Nous prouvons que l'algorithme est théoriquement correct et comparons sa performance avec les algorithmes les mieux connus, IAMB [12], PCMB [13] et PC [14]. Des expériences de simulations en utilisant des données générées de réseaux bayésiens connus, à savoir un réseau de petite envergure, Asia, contenant huit noeuds; deux réseaus de moyenne envergure, Alarm et PolyAlarm de 37 noeuds, et deux réseaux de plus grande envergure, Hailfinder contenant 56 noeuds et Test152 contenant 152 noeuds.

Les résultats démontrent qu'avec un nombre comparable d'observations, (1) IPC-MB obtient une reconnaissance nettement plus élevée que IAMB, jusqu'à 80% de réduction de distance (par rapport à un résultat parfait), (2) IPC-MB a une reconnaissance légèrement supérieure que PCMB et PC, et (3) IPC-MB nécessite jusqu'à 98% moins de tests conditionnels que PC et 95% de moins que PCMB (le nombre de tests conditionnels représente la mesure de complexité calculatoire ici).

La seconde contribution de la thèse est un algorithme pour induire la topologie du RB constitué des variables de la CM. Lorsqu'une CM d'une variable cible forme un RB, ce réseau est alors considéré comme un classificateur, nommé une Couverture de Markov de Classification (MBC).

L'algorithme a été nommé IPC-MBC sur la base du premier algorithme, IPC-MB. À l'instar de IPC-MB, l'algorithme IPC-MBC effectue une série de recherches locales pour éliminer les faux-négatifs, incluant les noeuds et les arcs. Cependant, sa complexité est supérieure et requiert des ressources calculatoires plus importantes que IPC-MB. Nous prouvons que IPC-MB est théoriquement et effectuons des études empiriques pour comparer sa performance calculatoire et de reconnaissance par rapport à PC seul et PC combiné à IPC-MB (c.-à-d. l'induction de la structure du RB avec l'algorithme PC seul et avec PC appliqué sur le résultat de IPC-MB). Les mêmes données que pour les expériences de simulation de IPC-MB sont utilisées. Les résultats démontrent que IPC-MBC combiné à IPC-MB et que PC combiné à IPC-MB sont tous deux plus efficaces que PC seul en termes de temps de complexité calculatoires, fournissant jusqu'à 95% de réduction du nombre de tests conditionnels, sans pour autant avoir d'impact au plan du taux de reconnaissance.

ABSTRACT

Feature selection is a fundamental topic in data mining and machine learning. It addresses the issue of dimension reduction by removing non-relevant, or less relevant attributes in model building. For the task of classification, a major milestone for feature selection was achieved by Koller and Sahami [1]. Building upon the work of Pearl on Bayesian Networks (BN) [2], they proved that a Markov blanket (MB) of a variable is the optimal feature subset for class prediction.

Deriving the MB of a class variable given a BN is a trivial problem. However, learning the structure of a BN from data is known to be NP hard. For large number of variables, learning the BN is impractical, not only because of the computational complexity, but also because of the data size requirement that is one of the curses of high dimensionality feature spaces.

Hence, simpler topologies are often assumed, such as the Naive Bayes approach (NB) [5, 6], which is probably the best known one due its computational simplicity, requiring no structure learning, and also its surprising effectiveness in many applications despite its unrealistic assumptions. One of its extension, Tree-Augmented Naïve Bayes (TAN) [7] is shown to have a better performance than NB, by allowing limited additional dependencies among the features. However, because they make strong assumptions, these approaches may be flawed in general. By further relaxing the restriction on the dependencies, a BN is expected to show better performance in term of classification accuracy than NB and TAN [8]. The question is whether we can derive a MB without learning the full BN topology for the classification task. Let us refer to a MB for classification as a Markov Blanket Classifier, MBC. The MBC is expected to perform as well as the whole Bayesian network as a classifier, though it is generally much smaller in size than the whole network.

This thesis addresses the problem of deriving the MBC effectively and efficiently from limited data. The goal is to outperform the simpler NB and TAN approaches that rely on potentially invalid assumptions, yet to allow MBC learning with limited data and low computational complexity.

Our first contribution is to propose one novel algorithm to filter out non-relevant attributes of a MBC. From our review, it is known that there are at least nine existing published works on the learning of Markov blanket since 1996. However, there is no satisfactory tradeoff between

correctness, data requirement and time efficiency. To address this tradeoff, we propose the IPC-MB algorithm [9-11]. IPC-MB performs an iterative search of the parents and children given a node of interest. We prove that the algorithm is sound in theory, and we compare it with the state of the art in MB learning, IAMB [12], PCMB [13] and PC [14]. Experiments are conducted using samples generated from known Bayesian networks, including small one like Asia with eight nodes, medium ones like Alarm and PolyAlarm (one polytree version of Alarm) with 37 nodes, and large ones like Hailfinder (56 nodes) and Test152 (152 nodes). The results demonstrate that, given the same amount of observations, (1) IPC-MB achieves much higher accuracy than IAMB, up to 80% reduction in distance (from the perfect result), (2) IPC-MB has slightly higher accuracy than PCMB and PC, (3) IPC-MB may require up to 98% fewer conditional independence (CI) tests than PC, and 95% fewer than PCMB. Given the output of IPC-MB, conventional structure learning algorithms can be applied to recover MBC without any modification since the feature selection procedure is transparent to them. In fact, the output of IPC-MB can be viewed as the output of general feature selection, and be employed further by all kinds of classifier. This algorithm was implemented by the author while working at SPSS and shipped with the software Clementine 12 in 2007.

The second contribution is to extend IPC-MB to induce the MBC directly without having to depend on external structure learning algorithm, and the proposed algorithm is named IPC-MBC (or IPC-BNC in one of our early publication) [15]. Similar to IPC-MB, IPC-MBC conducts a series of local searches to filter out false negatives, including nodes and arcs. However, it is more complex and requires greater computing resource than IPC-MB. IPC-MBC is also proved sound in theory. In our empirical studies, we compare the accuracy and time cost between IPC-MBC, PC and IPC-MB plus PC (i.e. structure learning by PC on the features output by IPC-MB), with the same data as used in the study of IPC-MB. It is observed that both IPC-MBC and IPC-MB plus PC are much more time efficient than PC, with up to 95% saving of CI tests, but with no loss of accuracy. This reflects the advantage of local search and feature selection respectively.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

D	Observations
$\mathbf{U}$	All attributes as contained in D
$\mathbb{G}$	Graph. Here, it is used to refer the directed acyclic graph of one Bayesian network, so $\mathbb{G} = \{\mathbf{E}, \mathbf{V}\}$, where $\mathbf{E}$ is the set of nodes, and $\mathbf{V}$ is the set of directed arcs
X	Variable or attribute
$\mathbf{X}$	Variable set or attribute vector, $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$
$\lvert\mathbf{X}\rvert$	Cardinality of set or, size of vector
T	Target variable, or dependent variable. $T \in \mathbf{U}$
ε	Significance value, or threshold value
ϕ	Empty set, or {}
$\mathbf{U}\backslash\{X\}$	Variable set $\mathbf{U}$ excluding X
BN	The concept of Bayesian network, by default over $\mathbf{U}$.
$\mathbb{BN}$	The structure of one Bayesian network, so it is a directed acyclic graph.
$\mathbf{Pa}_X$	Parents of X in $\mathbb{BN}$
$\mathbf{Ch}_X$	Children of X in $\mathbb{BN}$
$\mathbf{Sp}_X$	Spouses of X in $\mathbb{BN}$
$\mathbf{PC}_X$	Parents and children of X in $\mathbb{BN}$
$\mathbf{Des}_X$	Descendants of X in $\mathbb{BN}$
$\mathbf{PCD}_X$	$\mathbf{PCD}_X = \mathbf{PC}_X \cup \mathbf{Des}'_X$, where $\mathbf{Des}'_X \subseteq \mathbf{Des}_X$
$\mathbf{ND}_X$	Non-Descendants of of X in $\mathbb{BN}$
$\mathbf{MB}_X$	Markov blanket of X
$\mathbf{PC}_X^C$	Candidate parents and children of X
$\mathbf{Sp}_X^C$	Candidate spouses of X

$\mathbf{MB}_X^C$	Candidate Markov blanket of X
$\mathbb{MBC}_X$	Markov blanket classifier of X. It is the partial Bayesian network over variables $\mathbf{MB}_X \cup \{X\}$, so it is a directed acyclic graph as well
$\mathbb{BNC}_X$	Bayesian network classifier of X, another name of $\mathbb{MBC}_X$
$I(X, Y\|\mathbf{Z})$	Variable X is independent with Y given $\mathbf{Z}$
d-sep$(X, Y\|\mathbf{Z})$	X is d-separated from Y by node set $\mathbf{Z}$
$\sim I(X, Y\|\mathbf{Z})$	Variable X is NOT independent with Y given $\mathbf{Z}$
$I_D(X, Y\|\mathbf{Z})$	Statistical measure of (in)dependency between X and Y given $\mathbf{Z}$ with data D
iff.	If and only if, can be represented with ↔ or ⇔
$X - Y$	X is connected with Y in $\mathbb{G}$
$X \to Y$	X is connected with Y in $\mathbb{G}$, and the edge is pointing to Y
CI	Conditional independence
KS	Koller and Sahami's algorithm on the induction of $\mathbf{MB}_X$
GS	Grow-Shrink algorithm for the induction of $\mathbf{MB}_X$
IAMB	Incremental Association Markov Blanket
HITON-PC	One algorithm proposed to learn Markov blanket
/MB	
MMPC	Max-Min Markov boundary algorithm
/MB	
PCMB	Parents and Children based Markov Blanket algorithm
PC	PC algorithm, one classical method for the structure learning of Bayesian network
TAN	Tree-Augmented Naive Bayes. It can be viewed as a special Bayesian network, so it is a directed acyclic graph as well. If it refers to a specific model structure, we use $\mathbb{TAN}$ as the representation.

Chapitre 1 **INTRODUCTION**

1.1 Feature selection

Prediction, or classification, on one particular attribute in a set of observations is a common data mining and machine learning task. To analyze and predict the value of this attribute, we need to first ascertain which of the other attributes in the domain affect it. This task is frequently referred to as the *feature selection* problem. A solution to this problem is often non-trivial, and can be infeasible when the domain is defined over a large number of attributes.

In 1997, when a special issue of the journal of Artificial Intelligence on relevance, including several papers on variable and feature selection, was published, few domains explored used more than 40 features [16, 17]. The situation has changed considerably in the past decade: domains involving more variables but relatively few training examples are becoming common [12, 13, 18]. Therefore, feature selection has been an active research area in the pattern recognition, statistics and data mining communities. The main idea of feature selection is to select a subset of input variables by eliminating features with little or no predictive information, but without sacrificing the performance of the model built on the chosen features. It is also known as variable selection, feature reduction, attribute selection or variable subset selection. By removing most of the irrelevant and redundant features from the data, feature selection brings many potential benefits to us:

- Alleviating the effect of the curse of dimensionality to improve prediction performance;
- Facilitating data visualization and data understanding, e.g. which are the important features and how they are related with each other;
- Reducing the measurement and storage requirements;
- Speeding up the training and inference process;
- Enhancing model generalization.

A principled solution to the feature selection is to determine a subset of features that can render of the rest of the features independent of the variable of interest [1, 12, 13]. From a theoretical

perspective, it can be shown that optimal feature selection for supervised learning problems requires an exhaustive search of all possible subsets of features, the complexity of which is known as exponential function of the size of whole features. In practice, the target is demoted to a satisfactory set of features instead of an optimal set due to the lack of efficient algorithms.

Feature selection algorithms typically fall into two categories: Feature Ranking and Subset Selection. Feature Ranking ranks all attributes by a metric and eliminates those that do not achieve an adequate score. Selecting the most relevant variables is usually suboptimal for building a predictor, particularly if the variables are redundant. In other words, relevance does not imply optimality [17]. Besides, it has been demonstrated that a variable which is irrelevant to the target by itself can provide a significant performance improvement when taken with others [17, 19].

Subset selection, however, evaluates a subset of features that together have good predictive power, as opposed to ranking variables according to their individual predictive ability. Subset selection essentially divides into *wrappers*, *filters* and *embedded* [19].

In the wrapper approach, the feature selection algorithm conducts a search through the space of possible features and evaluates each subset by utilizing a specific modeling approach of interest as a black box [17], e.g. Naïve Bayes or SVM . For example, a Naïve Bayes model is induced with the given feature subset and assigned training data, and the prediction performance is evaluated using the remaining observations available. By iterating the training and cross-validation over each feature subset, wrappers can be computationally expensive and the outcome is tailored to a particular algorithm [17].

Filter is a paradigm proposed by Kohavi and John [17], and it is similar to wrappers in the search approach. A filter method computes a score for each feature and then select features according to their scores. Therefore, filters work independently of the chosen predictor. However, filters have the similar weakness as Feature Ranking since they imply that irrelevant features (defined as those with relatively low scores) are useless though it is proved not true [17, 19].

Embedded methods perform variable selection in the process of training and are usually specific to given learning algorithms. Compared with wrappers, embedded methods may be more efficient in several respects: they make better use of the available data without having to split the training data into a training and validation set; they reach a solution faster by avoiding retraining

a predictor from scratch for every variable subset to investigate [19]. Embedded methods are found in decision trees such as CART, for example, which have a built-in mechanism to perform variable selection [20].

1.2 Classification benefits from feature selection

In the classic supervised learning task, we are given a training set of labeled fixed-length feature vectors, or instances, from which to induce a classification model. This model, in turn, is used to predict the class label for a set of previously unlabeled instances. While, in a theoretical sense, having more features should give us more discriminating power, the real-world provides us with many reasons why this is not generally the case.

Foremost, many induction methods suffer from the curse of dimensionality. That is, as the number of features in an induction increases, the time requirements for an algorithm grow dramatically, sometimes exponentially. Therefore, when the set of features in the data is sufficiently large, many induction algorithms are simply intractable. This problem is further exacerbated by the fact that many features in a learning task may either be irrelevant or redundant to other features with respect to predicting the class of an instance. In this context, such features serve no purpose except to increase induction time.

Furthermore, many learning algorithms can be viewed as performing (a biased form of) estimation of the probability of the class label given a set of features. In domain with a large number of features, this distribution is very complex and of high dimension. Unfortunately, in the real world, we are often faced with the problem of limited data from which to induce a model. This makes it very difficult to obtain good estimates of the many parameters. In order to avoid over-fitting the model to the particular distribution seen in the training data, many algorithms employ the Occam's Razor [13] principle to build as simple a model as possible that still achieves some acceptable level of performance on the training data. This guide often leads us to prefer a small number of relatively predictive features over a large number of features.

If we could reduce the set of features considered by the algorithm, we can therefore serve two purposes. We can considerably decrease the running time of the induction algorithm, and we can increase the accuracy of the resulting model. In light of this, effort has been put on the issue of feature subset selection in machine learning as we mentioned in last section.

1.3 Bayesian Network, Markov blanket and Markov blanket classifier

Let **X** be the set of features, and T as the target variable of interest. **U** is used to refer our problem domain, and it is composed of **X** and T, i.e. $\mathbf{U} = \mathbf{X} \cup \{T\}$. A Markov blanket of T is any subset of **X** that renders T statistically independent from all the remaining attributes (see **Definition 7.4**). Koller and Sahami [1] first showed that the Markov blanket of a given target is the theoretically optimal set of attributes to predict its class value. If the probability distribution of $\mathbf{U} = \mathbf{X} \cup \{T\}$ can be faithfully (see **Definition 1.3**) represented by a Bayesian network (BN, see **Definition 1.1**) over **U**, then the Markov blanket of T is unique, just equal to its Markov boundary(see **Definition 7.4**), and it consists of the union of the parents, children and spouses of T in the corresponding BN [2]. Besides, the partial Bayesian network over the Markov blanket of T plus T itself is called Markov blanket classifier, or Bayesian network classifier (see **Definition 1.5**). Figure 1-1 illustrates a Bayesian network, Markov blanket of T and Markov blanket classifier with T as the target (or class).

Definition 1.1 (Bayesian Network) A Bayesian network consists of a directed acyclic graph (DAG) $\mathbb{G}$ an a set of local distributions. $\mathbb{G}$ is composed of nodes **V** and edges **E**, i.e. $\mathbb{G} = \{\mathbf{V}, \mathbf{E}\}$.

Definition 1.2 (Conditional Independence) Two sets of variables, **X** and **Y**, are said to be conditionally independent given some set of variables **Z** if, for any assignment of values **x**, **y** and **z** to the variables **X** , **Y** and **Z** respectively, $P(\mathbf{X} = \mathbf{x}|\mathbf{Z} = \mathbf{z}, \mathbf{Y} = \mathbf{y}) = P(\mathbf{X} = \mathbf{x}|\mathbf{Z} = \mathbf{z})$. That is, **Y** gives us no information about **X** beyond what is already in **Z**. We use $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ in the remaining text to denote this conditional independence relationship.

Definition 1.3 (Faithfulness Condition) A Bayesian Network $\mathbb{G}$ and a joint distribution P are faithful to one another iff. every conditional independence entailed by the graph $\mathbb{G}$ and the Markov Condition is also present in P [2].

Definition 7.4 (Markov blanket) A Markov blanket of an attribute $T \in \mathbf{U}$ is any subset **F** of $\mathbf{U}\backslash\{T\}$ for which T is conditionally independent with $\mathbf{U}\backslash\mathbf{F}\backslash\{T\}$ given the values of **F**. A set is called a Markov boundary of T if none of its proper subsets satisfy this condition.

Definition 1.5 (Markov blanket classifier) Given a Bayesian network $\mathbb{G}$ over the target variable T and attributes $\{X_i\}$, the partial DAG over $T \cup \mathbf{MB}_T$ is called the Markov Blanket Classifier, or Bayesian Network Classifier about T, and denoted as $\mathbb{MBC}_T$ or $\mathbb{BNC}_T$.

Definition 1.6 (Markov Condition) Given the value of parents, X is conditionally independent with all its non-descendants, denoted as $\mathbf{ND}_X$, excluding its parents $\mathbf{Pa}_X$, i.e. $I((X, \mathbf{ND}_X \setminus \mathbf{Pa}_X | \mathbf{Pa}_X,)$.

Theorem 1.1 If a Bayesian network $\mathbb{G}$ and a joint distribution P are faithful to one another, then for every attribute $T \in \mathbf{U}$, the Markov blanket of T is unique and is the set of parents, children and spouses of T.

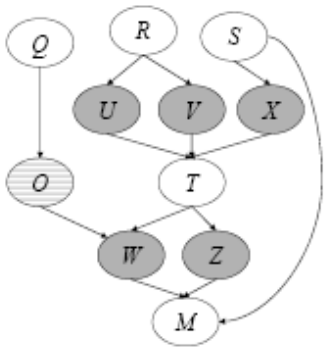


Figure 1-1: An example of a Bayesian network. The parents and children of T are the variables in gray, while $\mathbf{MB}_T$ additionally includes the textured-filled variable O. The partial network over $\mathbf{MB}_T$ and T are the Markov blanket classifier about T as class.

Let $\mathbf{Pa}_T$, $\mathbf{Ch}_T$ and $\mathbf{Sp}_T$ denote the parents, children and spouses of T respectively, the Markov blanket of T, denoted as $\mathbf{MB}_T$, then is the union of $\mathbf{Pa}_T$, $\mathbf{Ch}_T$ and $\mathbf{Sp}_T$ (see **Theorem 1.1**), i.e. $\mathbf{MB}_T = \mathbf{Pa}_T \cup \mathbf{Ch}_T \cup \mathbf{Sp}_T$ for short. Given this knowledge, $\mathbf{MB}_T$ of any $T \in \mathbf{U}$ is easily to be obtained if the Bayesian network over $\mathbf{U}$ is known. However, having to learn the Bayesian Network $\mathbb{G}$ in order to learn $\mathbf{MB}_T$ can be painfully time consuming [21]. Hence, how to learn $\mathbf{MB}_T$ but without having to learn the BN first became the goal of many who are interested to apply Markov blanket as feature selection.

Bayesian network, Markov blanket and Markov blanket classifier concepts are closely related given the faithfulness assumption. They will be frequently mentioned in the remaining text since our goals are efficient learning of Markov blanket and Markov blanket classifier. Chapitre 2 to

Chapitre 5 are about the learning of Markov blanket given a target of interest. The Markov blanket is important because

- It is the optimal feature subset for the prediction of T, and feature selection is an important data preprocessing step for most machine learning and data mining tasks;
- It is closely related to the Markov blanket classifier since the later is just a DAG over $\mathbf{MB}_T$ and T. Understanding this concept well will be helpful to understand our work on Markov blanket classifier;
- Given the Markov blanket of T, all existing structure learning algorithms of Bayesian network are applicable to induce the Markov blanket classifier. Since the feature space is greatly reduced, the remaining structure learning is expected to be much more efficient than using all features directly;
- Our algorithm for inducing the Markov blanket classifier of T is an extension of our algorithm on the induction of Markov blanket.

We return to the concept of Markov blanket classifier Chapitre 6.

1.4 KS and related algorithms

Following Koller and Sahami's work (KS), many others also realized that the principled solution to the feature selection problem is to determine a subset of features that can render the rest of all other features independent of the variable of interest [12, 13, 18, 21, 22]. Based on the findings that the full knowledge of $\mathbf{MB}_T$ is enough to determine the probability distribution of T and that the values of all other variables become superfluous, we normally can have a much smaller group of variables in the final classifier, reducing the complexity of learning and resulting with a simpler model, but without sacrificing classification performance [2, 3, 4].

Although Koller and Sahami theoretically proved that Markov blanket is the optimal feature subset for predicting the target, the algorithm as proposed by them for inducing $\mathbf{MB}_T$ is approximate, guaranteeing no correct outcome. There are several attempts to make the induction more effective and efficient, including GS (Grow-Shrink) [23, 24], IAMB (Iterative Associative Markov Blanket) and its variants [12, 18, 22], MMPC/MB (Max-Min Parents and

Children/Markov Blanket) [21], HITON-PC/MB[25], PCMB(Parent-Child Markov Blanket learning) [13] and our own work [10, 11, 26, 27], IPC-MB (Iterative Parents-Children based search of Markov Blanket), which will be discussed later. To our best knowledge, this list contains all the published primary algorithms. In next chapter, we will review these MB local learning algorithms in terms of theoretical and practical considerations, based on our experience gained from both academic research and industry implementation.

1.5 Motivation, contributions and overall structure

This project was initiated during my time in SPSS® (http://www.spss.com, acquired by IBM in 2009), where they needed a component of Bayesian Network for classification on widely deployed Clementine® (http://www.spss.com/software/modeling/modeler/, now named as PASW Modeler®). The greatest merit of a Bayesian Network is that its graphical model allows us to observe the relations of the variables involved, which is very important for diagnosis application. However, this component is designed primarily for classification, i.e. predicting the state of some target variable given input features, instead of general modeling. Regarding this goal, Pearl and Koller's works tell us that only the Markov blanket is effective in the prediction, which means that the partial Bayesian Network over the target and its Markov blanket is enough. This partial Bayesian Network is called Markov Blanket Classifier (MBC) or Bayesian Network Classifier (BNC) by us, to distinguish it from the whole Bayesian Network. It has all the merits of a general Bayesian Network, but it is "customized" for classification. In a naïve way, we can induce the Bayesian Network over all input variables first, and then extracting the MBC becomes trivial. This is possible and it requires no extra research effort, all existing conventional algorithms for the structure learning of Bayesian Network are there for reference. However, the learning of Bayesian network is known as an NP-complete problem, and the complexity grows exponentially in term of the number of inputs and the number of states of each individual input [3]. Therefore, the goal is to induce the MBC directly without having to learn the Bayesian Network first.

Koller and Sahami opened a new window, and many more fruitful studies have been done, along with many published outcomes. Given a bag of features **U**, these algorithms allow the induction of $\mathbf{MB}_T$ without requiring to know the Bayesian Network over **U** in advance. With $\mathbf{MB}_T$, the problem space generally is greatly reduced in dimension; besides, all existing algorithms for the structure learning of Bayesian Network are applicable, and they are expected to yield the

Bayesian Network structure over $\mathbf{MB}_T$. More importantly, due to the feature selection, all conventional structure learning algorithms are expected to solve larger scale of problems given the same computing resource. However, our review and experiments with all published algorithms on Markov blanket induction indicate that none of them was ideal, at least in the early 2007. Some of them may not yield the correct result; some may be efficient in time, but not data (or sample) efficient, which means that it requires large amount of data to produce satisfactory result; some of them may be quite data efficient, but quite poor in time efficiency.

Our first contribution is to propose a competitive algorithm for the local learning of Markov blanket. It is named as IPC-MB [10, 11, 27] since it is built on a series of iterative discovery of parents and children. IPC-MB is proved correct, and it is shown as much more data efficient than IAMB, and much more time efficient than PCMB, two well known algorithms for inducing $\mathbf{MB}_T$. As compared with PC [14, 28], one most known algorithm for the structure learning of BN, IPC-MB demonstrates obvious advantage as one requiring only local search, achieving great gain in time efficiency. IPC-MB was designed by myself for the induction of Markov blanket in the Bayesian Network component and implemented in Clementine® in 2007.

The second contribution extended IPC-MB to get IPC-MBC, which allows us to get the target Markov blanket classifier via efficient local search. It is called IPC-MBC since it also depends on the iterative discovery of parents and children, but it is more complex than IPC-MB because that it cares of not only to find $\mathbf{MB}_T$ but also the links existing among nodes of $\mathbf{MB}_T \cup \{T\}$. In our experiments, we compare IPC-MBC with not only PC, but IPC-MB+PC which calls IPC-MB to do feature selection first, and then depends on PC to finish the structure learning over $\mathbf{MB}_T$. The results show that although they have close performance on accuracy, IPC-MBC and IPC-MB+PC are much more time efficient than PC. Therefore, they are expected with better scalability.

So, we started with the problem of feature selection in classification application, and reviewed the family of algorithms on inducing Markov blanket. IPC-MB was proposed to compete with all existing similar ones, with exciting relative performance gained. Then, we went further to propose two effective and efficient algorithms for learning Markov blanket classifier, making full use of the knowledge and experience gained. In addition, we also study the combination of IPC-MB plus PC, and the results indicate that feature selection by IPC-MB not only greatly reduces the complexity of structure learning, but the overall timing cost. All these parts actually are

closely related. Efficiency, especially the data or sample efficiency, was emphasized all along the project since we always view the practical value as a very important evaluation criterion.

The remaining chapters of the thesis are organized as follows:

- A thorough review regarding the algorithms on learning Markov blanket is done in Chapter 2, which allows us to have in mind a comprehensive map about the existing work;

- In Chapter 3, a novel algorithm, called IPC-MB, for efficient learning of Markov blanket is proposed, including its motivation, specification, proof, complexity analysis and more discussion. It is categorized as local learning since it enables us to find the Markov blanket of target without having the whole Bayesian network known first, and it is expected to be the most data efficient among similar works, which is critical for algorithms built on statistical tests;

- Then, in Chapter 4, a series of empirical studies with data sampled from classical real networks are presented to give a comparison between IPC-MB and existing classical work, including IAMB, PCMB and PC algorithms, in term of accuracy, time and data efficiency. Besides, necessary implementation details are covered to make the results reproducible;

- A comprehensive trade-off analysis discussion about IAMB, PCMB, IPC-MB and PC is made in Chapter 5, including theoretical assumption, search strategy, data efficiency, time efficiency, potential scalability, information induced and implementation issues. All these factors are important for practical usage, so the discussion is believed valuable reference for applicants as well as researchers who are interested on this topic;

- In Chapter 6, we further propose an algorithm to learn the Markov blanket classifier without having to learn the whole Bayesian network first. It's called IPC-MBC, and it built on our knowledge and experience gained on previous work, especially IPC-MB. Experimental study is conducted over PC, IPC-MB+PC (which depends on IPC-MB to realize feature reduction first, then apply PC algorithm to finish the structure learning) and IPC-MBC with real networks, and the results indicate that both IPC-MB+PC and IPC-MBC achieve the similar accuracy as PC, but with much less cost on computing resource. With structure ready, the parameter learning is trivial, hence it is not covered in our discussion;

- Chapter 7 is a conclusion of the whole thesis as well as perspectives of our works.

Chapitre 2 REVIEW OF ALGORITHMS FOR MARKOV BLANKET LEARNING

Since Koller and Sahami's work in 1996 [1], there have been several efforts to make the learning procedure more efficient and effective. In this chapter, we will briefly review those known published works, including KS, GS, IAMB and its variants, MMPC/MB, HITON-PC/MB and PCMB. Because all these algorithms, including our own work, require faithfulness assumption (KS is an exception since it does not require this assumption), and depend on statistical (in)dependence test, we first discuss these two concepts in Section 2.1 and Section 2.2 respectively. Sections 2.3 to 2.8 are contributed for reviewing of those known published works.

2.1 Faithfulness Assumption

Faithfulness (see **Definition 1.3**) is an important concept that can be traced back to Pearl's work on Bayesian network in 1988 [2], and it is the most critical assumption as required by algorithms covered in the discussion here, including our own work but with KS as an exception. In its original texts [2, 29], Pearl et al. explained that, with the assumption of faithfulness, every distribution has a unique causal model (up to equivalence), as long as there are no hidden variables. This uniqueness follows from the fact the structural constraints that an underling DAG imposes upon the probability distribution are equivalent to a finite set of conditional independence relationships asserting that, given its parents, each variable is conditionally independent of all its non-descendents.

As we mentioned in last chapter, with this assumption, the Markov blanket also becomes unique, and is composed of the target's parents, children and spouses. Therefore, faithfulness builds a connection between probability distribution and graph structure. In the future discussion, we will demonstrate how PCMB and our work, IPC-MB, make use of this topology to increase the data efficiency which is known as the most disadvantage of GS, IAMB and their variants.

Lucky enough, the vast majority of distributions are faithful in the sample limit. Besides, for a number of different parametric families, the set of parameters that lead to violations of the faithfulness asssumption are Lebesgue measure 0 [28, 30].

2.2 Statistical dependence and independence

All algorithms covered here depend on asking for the true of independence relationships of the form:

$$I(X, Y|\mathbf{Z})$$

where **Z** is a subset of variables excluding X and Y. It can work with any source providing this kind of information. If we have a data set, this is answered by means of statistical tests of independence.

Among those works covered in this project, KS and IAMB employ cross entropy to measure the dependency, while the others choose Pearson's conditional independence χ^2 or G^2 test [31]. We would like to introduce them briefly respectively here.

2.2.1 Cross-entropy

If X and Y are random variables with joint probability distribution P, the cross entropy between them is defined as:

$$CE(X, Y) = \sum_{x,y} P(x, y) log\left(\frac{P(x, y)}{P(x)P(y)}\right)$$

Given three variables X, Y and Z, the cross entropy of X and Y given Z defined as:

$$CE(X, Y|Z) = \sum_{Z} P(z) \sum_{x,y} P(x, y|z) log\left(\frac{P(x, y|z)}{P(x|z)P(y|z)}\right)$$

This value is also called the ***mutual information***. It can be analogously defined when Z is a set of variables. It verifies similar properties to unconditional entropy, and it measures the degree of dependence of X and Y given Z. In particular, it is equal to 0.0 when this conditional independence is verified.

2.2.2 Pearson's Chi-Square test

Pearson's chi-square (χ^2) is the best-known of several chi-square tests, statistical procedures whose results are evaluated by reference to the χ^2 distribution. It can be used to access two types of comparison: tests of goodness of fit and tests of independence.

The χ^2 statistic is calculated by finding the difference between each observed and theoretical frequency, denoted as O and E respectively, for each possible outcome, squaring them, dividing each by the theoretical frequency, and taking the sum of the results. A second important part of determining the test statistic is to define the degrees of freedom of the test.

In the test of independence, an "observation" consists of the values of two outcomes and the null hypothesis is that the occurrence of these outcomes is statistically independent. Each observation is allocated to one cell of a two-dimensional array of cells according to the values of the two outcomes. If there are r rows and c columns, and totally n cells in the table, the theoretical frequency for a cell, given the hypothesis of independence, is

$$E_{i,j} = \frac{\sum_{k=1}^{c} O_{i,k} \sum_{k=1}^{r} O_{k,j}}{n} \tag{1.1}$$

and fitting the model of "independence" reduces the number of degrees of freedom by $d = r + c - 1$. The value of the test-statistic is

$$\sum_{i=1}^{r} \sum_{j=1}^{c} \frac{\left(O_{i,j} - E_{i,j}\right)^2}{E_{i,j}} \tag{1.2}$$

The number of degrees of freedom is equal to the number of cells rc, minus the reduction in degrees of freedom, d, which reduces to $(r-1)(c-1)$.

For the test of independence, the χ^2, a probability > 0.05 is commonly interpreted as justification for rejecting the null hypothesis that the row variable is unrelated to the column variable.

The χ^2 test requires minimal cell sizes. A common rule is 5 or more in all cells of a 2-by-2 table, and 5 or more in 80% of cells in larger tables, but no cells with zero count. When this assumption is not met, Yates' correction is applied (refer to next section).

2.2.3 Chi-Square test with Yates correction

The approximation to the χ^2 distribution breaks down if expected frequencies are too low. It will normally be acceptable so long as no more than 10% of the events have expected frequencies

below 5. Where there is only 1 degree of freedom, the approximation is not reliable if expected frequencies are below 10. In this case, a better approximation can be obtained by reducing the absolute value of each difference between observed and expected frequencies by 0.5 before squaring, and this is called Yates' correction for continuity. The following is Yates' corrected version of Pearson's χ^2 statistic:

$$\sum_{i=1}^{r}\sum_{j=1}^{c}\frac{\left(\left|O_{i,j}-E_{i,j}\right|-0.5\right)^2}{E_{i,j}} \tag{1.3}$$

2.2.4 G^2 Test

In cases where the expected value, E, is found to be small (indicating either a small underlying population probability, or a small number of observations), the normal approximation of the multinomial distribution can fail, and in such cases it is found to be more appropriate to use the G^2, a likelihood ratio-based test statistic.

The commonly used chi-squared tests for goodness of fit to a distribution and for independence in contingency tables are in fact approximations of the log-likelihood ratio on which the G^2 tests are based. This approximation was developed by Karl Pearson because at the time it was unduly laborious to calculate log-likelihood ratios. Due to the introduction of computers, however, G^2 tests are coming into increasing use.

The general formula for G^2 corresponding to equation (1.2) is

$$G = 2\sum_{i=1}^{r}\sum_{j=1}^{c} O_{i,j}\cdot ln\left(\frac{O_{i,j}}{E_{i,j}}\right) \tag{1.4}$$

where ln denotes the natural logarithm and the sum is taken over all non-empty cells. Given the null hypothesis that the observed frequencies are random sampling from a distribution with the given expected frequencies, the distribution of G is approximately that of χ^2, with the same number of degrees of freedom as in the corresponding χ^2 test.

For samples of a reasonable size, the G^2 test and the χ^2 test will lead to the same conclusions. However, the approximation to the theoretical χ^2 distribution for the G^2 test is better than for the

Pearson χ^2 tests in cases where for any cell $|O_i - E_i|/E_i > 1$, and in any such case the G^2 test should always be used [32].

2.2.5 Our Choice

In our implementation, the G^2 test is chosen based on the following knowledge gained from our own review and experimental studies:

1. It is preferred by similar research work [13, 21], and we want to produce a comparable results;
2. It produces better results than Pearson χ^2 test with or without Yates correction in our empirical studies, especially when sample size is relatively small.

Besides, in our implementation, a test $I_D(X, Y|\boldsymbol{Z})$ will be ignored if

$$\frac{N}{|X| \times |Y| \times \prod_{\mathrm{Z_i} \in \mathbf{Z}} |\mathrm{Z_i}|} \geq k \tag{1.5}$$

where N is the total number of observations available, $|X|$ and $|Y|$ are the number of value that X and Y can have respectively, and k is an empirical threshold value. This in-equation gives an empirical standard to decide if a test is reliable or not, that is the minimum average number of observations available in each cell of a contingency table should be at least k. In all our experiments we choose $k = 5$ because, as suggested by Agresti [31], this is the minimum average number of instances per cell for the G statistic to have χ^2 distribution, and it is applied by several similar published works like Fast-IAMB [22] and PCMB [13].

Note: (1) G^2 test is employed in the implementation of all algorithms studied in our experiments for fair comparison purpose; (2) In our implementations, $I_D(X, Y|\mathbf{Z}) \leq \varepsilon$ means that X and Y are conditionally dependent given $\mathbf{Z}$*, while $I_D(X, Y|\mathbf{Z}) > \varepsilon$ indicates that X and Y are conditionally independent given* $\mathbf{Z}$*.*

2.3 KS (Koller and Sahami's Algorithm)

In the following sections, we will review the algorithms introduced for deriving an MB, starting with the earlier ones and towards the most sophisticated and latter ones.

Koller and Sahami proposed a framework for optimal feature selection by measuring and minimizing the amount of predictive information lost during feature elimination [1]. They also proposed an approximate algorithm based on their theoretical model, and this algorithm is referred to as KS by many since then. KS is the first algorithm for feature selection to employ the concept of Markov blanket, and it accepts two parameters, (1) the number of variables to retain, i.e. the limit of the target Markov blanket, and (2) the maximum number of variables it is allowed to condition on. Both settings are useful to reduce the search space, but obviously, it is a heuristic and approximate algorithm, not always guaranteeing correct outcome.

Basically, KS is a filter algorithm which does not incur the high computational cost of conducting a search through the space of feature subsets as in the wrapper methods, and is therefore efficient for domains containing hundreds or even thousands of features. Although it is theoretically sound, the KS algorithm itself will not always produce correct outcomes. In this section, how KS works will be described with a little more detail.

Although this algorithm is simple and easy to implement, it is clearly suboptimal in many ways, particularly due to the very naïve approximations that it uses. Koller and Sahami also discussed some ways to possibly improve the result, and more importantly, they point out that increasing the size of the conditioning set would fragment the training set into small chunks, and result with a degradation on performance. Though it was noticed as early as in 1996, this problem was not conquered by latter algorithms, until the introduction of PCMB.

2.4 GS (Grow-Shrink)

The Grow-shrink (GS) [23, 24] algorithm was proposed to induce the structure of Bayesian network via the discovery of local neighbours, i.e. Markov blanket of each node. The GS algorithm actually contains two independent components, GSMB and GSBN. GSMB is responsible to induce the Markov blanket of a variable, and GSBN is employed to induce the whole Bayesian network by using the knowledge supplied by GSMB. Therefore, when we mention GS in the context of Markov blanket learning, in fact, it is called GSMB in the original literature. In this thesis, we will continue the usage of GS to refer to GSMB considering that no ambiguity will be introduced.

GS employs independence properties of the underlying network to discover parts of its structure, just like the SGS and PC algorithms [14, 28]. However, the design of GS enables it to addresses the two main shortcomings of the prior work which are preventing its use from becoming more widespread. These two disadvantages are: exponential execution time and proneness to errors in dependence tests used. The former problem is addressed in [23, 24] in two ways. One is by identifying the local neighbourhood of each variable in the Bayesian network as a pre-processing step in order to facilitate the recovery of the local structure around each variable in polynomial time under the assumption of bounded neighbourhood size. The second, randomized version goes one step further, employing a user-specified number of randomized tests (constant or logarithmic) in order to ascertain the same result with high probability. The second disadvantage of this research approach, namely proneness to errors, is also addressed by the randomized version, by using multiple data sets (if available) and Bayesian accumulation of evidence.

Although the concept of the Markov blanket is not new, GS is known as the first to explicitly use this idea to effectively limit unnecessary computation while inducing the underlying Bayesian network. GS(MB) itself is simple, and it proceeds in two phases: grow first, shrink secondly (see Figure 1-2).

Here, **U** denotes the complete set of variables. The idea behind the growing phase is simple: as long as the Markov blanket property of T is violated, i.e. there exists a variable in **U** that is dependent on T, it is added to **S** until there are no more such variables. In this process however, there may be some variables that were added to **S** that were really outside the blanket. Such variables would have been rendered independent from T at a later point when all Markov blanket nodes of the underlying Bayesian network were added to **S**. This observation necessitates the shrinking phase, which identifies and removes those variables. Finally, what is left in **S** is known the Markov blanket of T, $\mathbf{MB}_T$.

Theorem 1.2 Given the assumption of faithfulness and correct independence test, GS induces the correct Markov blanket [23, 24].

The algorithm is efficient, requiring only $O(n)$ conditional tests. One may minimize the number of tests in shrinking phase by heuristically ordering the variables in the loop of growing phase, for example by ascending mutual information or probability of dependence between X and Y. In [23, 24], Margaritis and Thrun also proposed one randomized version of GS algorithm to solve

problems involving large amount of variables or variables with many possible values. It requires manually defined parameter to reduce the number of conditional tests, so it cannot guarantee correct output, and it is ignored without further discussion.

```
GS(T: Target, D: Dataset, ε: Significance Value)
{
1.  MB_T^G ← {};
    //Growing phase
2.  repeat
3.      stillGrow ← false;
4.      for(∀X ∈ U\MB_T^G\{T}) do
5.          If (I_D(T, X|MB_T^G) ≤ ε) then
6.              MB_T^G ← MB_T^G ∪ {X};
7.              stillGrow ← true;
8.          end if
9.      end for
10. until stillGrow = false
    //Shrinking phase
11. repeat
12.     stillShrink ← false;
13.     for(∀X ∈ MB_T^G) do
14.         If (I_D(T, X|MB_T^G − {X}) > ε) then
15.             MB_T^G ← MB_T^G\{X};
16.             stillShrink ← true;
17.         end if
18.     end for
19. until stillShrink = fasle
20. return MB_T^G;
}
```

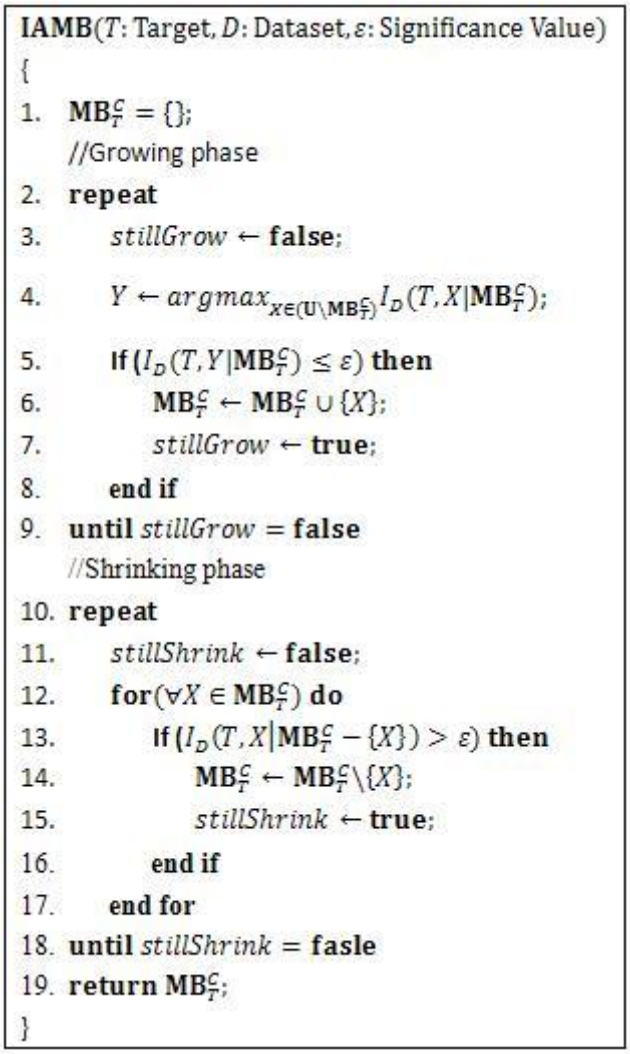
Figure 1-2: Grow-shrink (GS) algorithm.

2.5 IAMB and Its Variants

2.5.1 IAMB

Following KS and GS algorithms, Tsamardinos et al. proposed a series of algorithms for inducing the Markov blanket of a variable T of interest without having to learn the whole Bayesian network first. All of these works are based on same two assumptions as required by GS: 1) the

data are generated by processes that can be faithfully represented by BNs, and 2) there exist reliable statistical tests of conditional independence and measures of associations for the given variable distribution, sample size and sampling of the data.



IAMB(T: Target, D: Dataset, ε: Significance Value)

{

1. $\mathbf{MB}_T^G = \{\}$;

   //Growing phase

2. **repeat**
3. $stillGrow \leftarrow$ **false**;
4. $Y \leftarrow argmax_{X \in (\mathbf{U} \setminus \mathbf{MB}_T^G)} I_D(T, X | \mathbf{MB}_T^G)$;
5. **If** $(I_D(T, Y | \mathbf{MB}_T^G) \leq \varepsilon)$ **then**
6. $\mathbf{MB}_T^G \leftarrow \mathbf{MB}_T^G \cup \{X\}$;
7. $stillGrow \leftarrow$ **true**;
8. **end if**
9. **until** $stillGrow =$ **false**

   //Shrinking phase

10. **repeat**
11. $stillShrink \leftarrow$ **false**;
12. **for**$(\forall X \in \mathbf{MB}_T^G)$ **do**
13. **If** $(I_D(T, X | \mathbf{MB}_T^G - \{X\}) > \varepsilon)$ **then**
14. $\mathbf{MB}_T^G \leftarrow \mathbf{MB}_T^G \setminus \{X\}$;
15. $stillShrink \leftarrow$ **true**;
16. **end if**
17. **end for**
18. **until** $stillShrink =$ **fasle**
19. **return** $\mathbf{MB}_T^G$;

}

Figure 1-3: IAMB algorithm

The primary algorithm proposed by Tsamardinos et al. is called Incremental Association Markov Blanket, or IAMB (Figure 1-3). IAMB consists of two steps, a forward and a backward one, which actually is akin to the growing and shrinking phases in GS. This algorithm relies on an independence test, $I_D(\cdot)$, that is considered true (or succeeded) if it is smaller or equal than a threshold and false (failed) otherwise. It is important that $I_D(\cdot)$ is an effective test so that the set of candidate variables after Phase I is as small as possible for two reasons: one is time efficiency (i.e., do not spend time considering irrelevant variables) and another is sample efficiency (do not

require sample larger than what is necessary to perform conditional tests of independence). Since this step is based on the heuristic at line 5, some nodes not in $\mathbf{MB}_T$ may be added to $\mathbf{MB}_T^C$ as well. In Phase II (backward), we remove one-by-one the features that do not belong to $\mathbf{MB}_T$ by testing the whether a feature $X \in \mathbf{MB}_T$ is independent of T given the remaining $\mathbf{MB}_T^C$ (lines 10-18).

IAMB algorithm is structurally similar to GS algorithm, and follows the same two-phase structure. However, there is an important difference: GS may order the variables when they are considered for inclusion in phase I, according to their strength of association with T given the empty set [23, 24] (this appears in the discussion for better performance in the original text, but not in Figure 1-2). It then admits into $\mathbf{MB}_T^C$ the next variable in the ordering that is not conditionally independent from T given the current $\mathbf{MB}_T^C$. One problem with this heuristic is that when the $\mathbf{MB}_T^C$ contains spouses of T, the spouses are typically associated with T very weakly given the empty set and are considered for inclusion in the $\mathbf{MB}_T^C$ late in the first phase (associations between spouses and T are only through confounding/common descendant variables, thus they are weaker than those ancestors' associations with T). In turn, this implies that more false positives will enter $\mathbf{MB}_T^C$ during phase I and the conditional tests of independence will become unreliable much sooner than when using IAMB's heuristic. In contrast, conditioned on the common children, spouses may have strong association with T and, when using IAMB's heuristic, and enter the $\mathbf{MB}_T^C$ early.

2.5.2 InterIAMBnPC

Tsamardinos et al. recognized and pointed out explicitly that the smaller the conditioning test given a finite sample of fixed size, the more accurate are the statistical tests of independence and the measure of association [12, 18, 21]. In other words, to have a reliable decision given independence test of high degree, we need a large amount of instances for training. Though IAMB provides guarantees on correctness theoretically, it is only suited for the cases where the available sample size is large enough to perform condition independence tests as conditioned on the full $\mathbf{MB}_T$ or even larger set. Some variants are therefore proposed to decrease the critical requirement of data size, which just reflects the authors' emphasis on the practical value of their work.

InterIAMBnPC is one such variant aiming to further reduce the size of the conditioning sets [18]. It employs two methods for this goal: (1) it interleaves the growing phase of IAMB with the pruning phase attempting to keep the size of $\mathbf{MB}_T^C$ as small as possible during all steps of the algorithm's execution; (2) it substitutes the shrinking phase as implemented in IAMB with the PC algorithm instead [14], a Bayesian Network learning algorithm that determines directed edges between variables in a more sample-efficient manner.

Two other IAMB variants experimented in [12, 18] are InterIAMB and IAMBnPC which are similar to InterIAMBnPC but they employ only either interleaving the first two phases or using PC for the backward phase respectively. Considering that they have no fundamental difference compared to InterIAMBnPC, no more space is consumed for further introduction of these two algorithms.

2.5.3 Fast-IAMB

Fast-IAMB was proposed in 2005, and it is also built on the two assumptions: faithfulness and correct independence test [22]. Similar to GS and IAMB, Fast-IAMB contains a "growing" phase and a "shrinking" phase. During the growing phase of each iteration, it sorts the attributes that are candidates for admission to $\mathbf{MB}_T^C$ from most to least conditionally dependent, according to a heuristic function h (corresponding to $I_D(\cdot)$ in IAMB; it is mutual information in IAMB, but G^2 conditional statistical test here). Each such sorting step is potentially expensive since it involves the calculation of the G^2 test static between T and each member of $\mathbf{S}$. The key idea behind Fast-IAMB is to reduce the number of such tests by adding not one, but a number of attributes at a time after each reordering of the remaining attributes following a modification of the Markov blanket. Fast-IAMB speculatively adds one or more attributes of highest G^2 test significance without re-sorting after each modification as IAMB does, which (hopefully) adds more than one true member of the blanket. Thus, the cost of re-sorting the remaining attributes after each Markov blanket modification can be amortized over the addition of multiple attributes.

The question arises: how many attributes should be added to the blanket within each iteration? The following heuristic is used in [22]: dependent attributes are added as long as the conditional independence tests are reliable, i.e. there is enough data for conducting them. For this purpose, a numeric parameter k is used to denote the minimum average number of instances per cell of a

contingency table that should be present for a conditional independence test to be deemed reliable. Please refer to section 2.2.5 for the discussion of a reliable test and the choice of k.

Fast − IAMB(T: Target, D: Dataset, ε: Significance Value)
{
1. $\mathbf{MB}_T^C \leftarrow \{\}$;
2. $\mathbf{S} \leftarrow \{X | X \in \mathbf{U} \backslash \{T\}\ and\ I_D(T, X) \leq \varepsilon)$;
3. **while**($\mathbf{S} \neq \{\ \}$) **do**
4. $< X_1, \ldots, X_{|S|} > \leftarrow \mathbf{S}$ sorted according to I_D;
5. $insufficient\_data \leftarrow$ **false**;
   //Growing phase
6. **for**($i = 1$ to $|\mathbf{S}|$)**do**
7. **if**($\frac{N}{r_{Xi} \times r_T \times r_{MB(T)}}$) $\geq$ k **then**
8. $\mathbf{MB}_T^C \leftarrow \mathbf{MB}_T^C \cup \{X_i\}$;
9. **else**
10. $insufficient\_data \leftarrow$ **true**;
11. **go to** 14 /* insufficient data*/
12. **end if**
13. **end for**
    // Shrinking phase
14. $stillShrink \leftarrow$ **fasle**;
15. **for**($\forall X \in \mathbf{MB}_T^C$)**do**
16. **if**($I_D(T, X | \mathbf{MB}_T^C - \{X\})$ **then**
17. $\mathbf{MB}_T^C \leftarrow \mathbf{MB}_T^C \backslash \{X\}$;
18. $stillShrink \leftarrow$ **true**;
19. **end if**
20. **end for**
21. **if**($insufficien\_data =$ **true** and $stillShrink =$ **false**) **do**
22. **break**;
23. **else**
24. $\mathbf{S} \leftarrow \{X | X \in \mathbf{U} \backslash \mathbf{MB}_T^C \backslash \{T\}\ and\ I_D(T, X | \mathbf{MB}_T^C) \leq \varepsilon)\}$;
25. **end if**
26. **end while**
27. **return** $\mathbf{MB}_T^C$;
}

Figure 1-4: Fast-IAMB algorithm.

The authors of Fast-IAMB also answer explicitly one practical question that the authors of IAMB didn't mention in their work [18], namely what is to be done if the average number of instances

per cell for each remaining attribute is less than k? In this case, one has two choices: assume dependence or assume independence. While assuming dependence might seem to be the "safe" choice, in practice this would result in large blankets that are hard to justify and of little practical use. Therefore, independence is assumed in [22], which results in halting (Line 22, Figure 1-4) and returning the current blanket. This is followed in our implementation as well.

In conclusion, Fast-IAMB follows the previous work of GS and IAMB, especially Inter-IAMB by interleaving growing and shrinking. To realize a fast induction, greedy strategy is employed in growing by adding as many candidates as possible if allowed. Compared with IAMB, it emphasizes more the practical value of the algorithm, which is highly desired for practitioners. Although the authors declared it is fast and it is indeed demonstrated by their experiments relative to IAMB and Inter-IAMB, we consider that doubt remains about this point since more statistical tests are possibly expected in the shrinking phase if more false positive ones are added in the growing state.

2.6 MMMB (Max-Min Markov Boundary algorithm)

Starting with KS, and followed by much effort, several efficient algorithms to induce the Markov blanket given a target T of interest have been proposed. However, none of them ever make use of the underlying topology information to improve the efficiency, especially the data efficiency, given the faithfulness assumption. The Max-Min Markov Blanket (MMMB) algorithm is proposed here to improve data efficiency over previously known algorithms for inducing Markov blanket, because the sample requirements of MMMB depend on the connectivity and topology of the Bayesian network faithful to the data, but the others depend on the size of the learned Markov blanket.

2.6.1 Bayesian Network and Markov Blanket

Since the underlying topology information will possibly help to increase the performance of MB induction algorithms, we revisit theoretical considerations and introduce additional background knowledge about Bayesian networks. A Bayesian network is a graphical tool that compactly represents a joint probability distribution P over a set of random variables $\mathbf{U}$ using a directed acyclic graph (DAG) $\mathbb{G}$ annotated with conditional probability tables of the probability distribution of a node given any instantiation of its parents. The graph represents qualitative

information about the random variables (conditional independence properties), whiles the associated probability distribution, consistent with such properties, provides a quantitative description of how the variables relate to each other. An example of BN is shown in Figure 1-1. The probability distribution P and the graph $\mathbb{G}$ of a BN are connected by the Markov Condition property: a node is conditionally independent of its non-descendants, given its parents.

As its name indicates, DAG is formed by a collection of vertices and directed edges, each edge connecting one vertex to another, such that there is NO way to start at some vertex X and follow a sequence of edges that eventually loops back to X again [33]. Each DAG gives rise to a partial order $\leq$ on its vertices, where $X \leq Y$ if there exists a directed path from Y to X. $X \leq Y$, in fact, means that X is a descendant of Y, and its formal definition is given with **Definition 1.7** for later reference. Each DAG has a topological ordering, an ordering of the vertices such that the starting endpoint of every edge occurs earlier in the ordering than the ending endpoint of the edge. In general, this ordering is not unique; A DAG has a unique topological ordering if and only if it has a directed path containing all the vertices, in which case the ordering is the same as the order in which the vertices appear in the path [28, 33, 34].

Definition 1.7 (Descendant) Y is a descendant of X, if there exists a directed path from X to Y, but there exists no directed path from Y to X. The set of descendants of X is denoted with $\mathbf{Des}_X$ in the remaining text.

Definition 1.8 (Non-Descendant) Given all variable set $\mathbf{U}$, those other than descendants are known as non-descendants of X, denoted as $\mathbf{ND}_X$. $\mathbf{ND}_X = \mathbf{U} \backslash \mathbf{Des}_X$.

As mentioned above, if we know the Bayesian network over $\mathbf{U}$ in advance, it is trivial to get the $\mathbf{MB}_T$ of interest. The partial structure over $\{T\} \cup \mathbf{MB}_T$ is also a directed acyclic graph (DAG); recall that, for any $X \in \mathbf{U}$ and $X \in \mathbf{MB}_T$, it has to satisfy one of the two graphical constraints:

1. Either X is connected to T directly, more specifically $X \to T$ or $T \to X$, when X is parent or child of T; or,
2. X shares some common child(ren) Y with T, i.e. $X \to Y \leftarrow T$, when X is known as the spouse of T.

D-separation is the criterion that allows computation of the entailed independence in a Bayesian network from the Markov Condition [2]. D-separation is defined on the basis of blocked paths:

Definition 1.9 Collider node and Blocked path. A node X of a path p is a collider if p contains two incoming edges into X (e.g., W in Figure 1-1 is a **collider** in the path $O \rightarrow W \leftarrow T$). A path p from node X to node Y is **blocked** by a set of nodes **Z**, if any of the following is true: (1) There is a non-collider node on p that belongs to **Z**; (2) No collider nodes of p and none of their descendants belong in **Z**.

Definition 1.10 d-separation Two nodes X and Y are d-separated by **Z** if and only if every path from X to Y is blocked by **Z**, and it is denoted as $d - sep(X, Y|\mathbf{Z})$.

Theorem 1.3 If a Bayesian network G is faithful to a distribution P, then $d - sep(X, Y|\mathbf{Z}) \Leftrightarrow I(X, Y|\mathbf{Z})$, i.e. the conditional independence relation in P is equal to d-separation in $\mathbb{G}$.

With the theorem presented and the faithfulness assumption, the terms d-separation and conditional independence are used interchangeably in the rest of the article.

By performing independence tests and considering the d-separation relations they entail, one can reconstruct the Bayesian network corresponding to the distribution. This is the main idea behind constraint-based, or CI test-based, Bayesian network learning approaches [8, 14, 23, 24, 35, 36]. The following theorem in [14, 28] is foundational for both PC and MMMB algorithm here:

Theorem 1.4 If a Bayesian network $\mathbb{G}$ is faithful to a joint probability distribution P then:

1. There is an edge between the pair of nodes X and Y in $\mathbb{G}$ iff X and Y are conditionally dependent given any other set of nodes;
2. For each triplet of nodes X, Y and Z in $\mathbb{G}$ such that X and Y are adjacent to Z but X is not adjacent to Y, $X \rightarrow Z \leftarrow Y$ is a subgraph of $\mathbb{G}$ iff. X and Y are dependent conditioned on every other set of nodes that contains Z.

The first part of the theorem allows us to infer the existence of edges or not, and the second part to determine the known v-structure which actually allows us to determine the orientation of related arcs.

Given the faithfulness assumption, the Markov blanket of T, $\mathbf{MB}_T$, can be defined either probabilistically (as the minimal set conditioned on which every other node is independent is independent of T) or graph theoretically (as the set of parents, children, and spouses of T).

Definition 1.11 Markov Blanket (Probabilistic viewpoint) Given the faithfulness assumption, the Markov blanket of T, $\mathbf{MB}_T$, is a minimal set conditioned on which all other nodes are independent of T, i.e. $\forall X \in \mathbf{U} \backslash \mathbf{MB}_T \backslash \{T\}$, $I(X, T|\mathbf{MB}_T)$.

Definition 1.12 Markov Blanket (Graphical viewpoint) Given the faithfulness assumption, the Markov blanket of T, $\mathbf{MB}_T$, is identical to T's parents, children and children' parents (spouses), i.e. $\mathbf{MB}_T = \mathbf{Pa}_T \cup \mathbf{Ch}_T \cup \mathbf{Sp}_T$.

Before MMPC/MB, algorithms like IAMB and GS only depend on the (in)dependence property as derived from **Definition 1.11** to recognize positive as well as false positive ones, though the property as contained in **Definition 1.12** was known. Compared with previous works, MMPC/MB works in a different way. It is built on the basis of **Theorem 1.4**, and the induction of target $\mathbf{MB}_T$ is divided into the recognition of $\mathbf{PC}_T (= \mathbf{Pa}_T \cup \mathbf{Ch}_T)$ and $\mathbf{Sp}_T$ separately. It depends on a series of conditional independence tests, like $I(X, Y|\mathbf{Z})$, to decide if there exists edge between X and Y. Generally, $\mathbf{Z}$ is smaller than $\mathbf{MB}_T$, hence MMPC/MB finds a novel way to achieve better data efficiency than GS and IAMB. Actually, HITON, PCMB and our IPC-MB are all proposed on the basis on this important finding.

2.6.2 D-separation

Since d-separation is frequently referred during our proof, in this section, we step further to introduce how to determine if node X is d-separated from Y, which is equal to determine conditional independency given faithfulness assumption.

Bayesian networks encode the dependencies and independencies among variables. Under the causal Markov assumption, each variable in a Bayesian network is independent of its ancestors given the values of its parents [2], which permits us to infer some conditional independence relationships. For the general conditional independence in a Bayesian network, Pearl proposed a concept called d-separation [2]. D-separation, as short for direction-dependent separation, is a graphical property of Bayesian networks and has the following implication: If two sets of nodes $\mathbf{X}$ and $\mathbf{Y}$ are d-separated in Bayesian networks by a disjoint set $\mathbf{Z}$ (i.e. $\mathbf{X} \cap \mathbf{Y} \cap \mathbf{Z} = \emptyset$), the corresponding variable sets $\mathbf{X}$ and $\mathbf{Y}$ are independent given the variables in $\mathbf{Z}$. The definition of d-separation (**Definition 1.10**) tells us that $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by a disjoint set $\mathbf{Z}$ iff. every

undirected path between **X** and **Y,** i.e. $X - \cdots Z \cdots - Y$, is "blocked", where $X \in \mathbf{X}$, $Y \in \mathbf{Y}$ and $Z \in \mathbf{Z}$. The term "blocked" means:

- Either the connection through Z is "tail-to-tail" or "tail-to-head" and Z is instantiated, i.e. $Z \in \mathbf{Z}$; or
- The connection through Z is "head-to-head" and neither Z nor any of Z's descendants has received evidence, i.e. $Z \notin \mathbf{Z}$.

The graph patterns of "tail-to-tail (diverging)", "tail-to-head (serial)" and "head-to-head (converging or collider)" are shown as below (Figure 1-5):

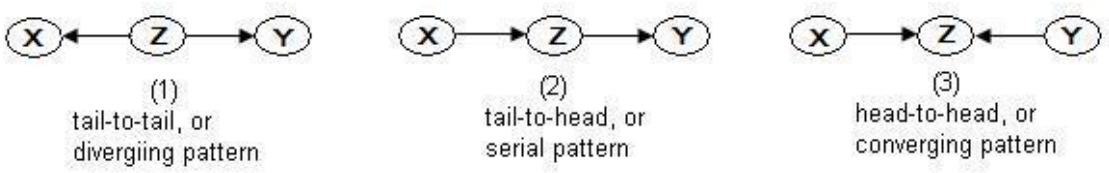


Figure 1-5: Three possible patterns about any path through a node in Bayesian network

With the definition of d-separation, and the three graphical patterns as demonstrated in Figure 1-5, we are interested in proving $I(T, \mathbf{U}\backslash\mathbf{MB}_T\backslash\{T\}|\mathbf{MB}_T)$ from the viewpoint of d-separation, i.e. $\mathbf{MB}_T$ d-separates T from $\mathbf{U}\backslash\mathbf{MB}_T\backslash\{T\}$, and $\mathbf{MB}_T$ is the minimal such set given faithfulness assumption.

Theorem 1.5 Given the faithfulness assumption, the minimal set of nodes which d-separates the node T from all other nodes is T's Markov blanket.

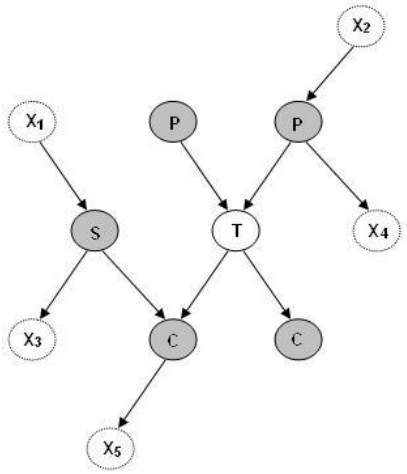


Figure 1-6: The Markov blanket of T (includes P(arents), C(hildren) and S(pouses)) d-separates all other nodes given faithfulness assumption.

Proof. $\mathbf{MB}_T$ *contains* T*'s parents, children and spouses, which are represented with grayed circles and denoted as* P, C *and* S *respectively in Figure 1-6. Those* $X \notin \mathbf{MB}_T$ *but are connected to* P, C *or* S *are represented by circles with dotted edge. In total, there are five possible such cases, denoted as* X_1, X_2, X_3, X_4 *and* X_5 *respectively, and we will explain how they are all d-separated from* T*, given the precondition that the whole* $\mathbf{MB}_T$ *are instantiated (denoted as grayed circles):*

1. $X_1 \to S \to C \leftarrow T$*: There exists serial (tail-to-head) pattern (*$X_1 \to S \to C$*) on this path, and* S *is instantiated , so this path is "blocked";*
2. $X_2 \to P \to T$*: This path is "blocked" due to the existing of serial(tail-to-head) pattern with* P *instantiated;*
3. $X_3 \leftarrow S \to C \leftarrow T$*: There exists diverging(tail-to-tai)l pattern (*$X_3 \leftarrow S \to C$*) on this path, and* S *is instantiated , so this path is "blocked";*
4. $X_4 \leftarrow P \to T$*: This path is "blocked" due to the existing of diverging(tail-to-tai)l pattern with* P *instantiated;*
5. $X_5 \leftarrow C \leftarrow T$*: This path is "blocked" is due to the existing of serial(tail-to-head) pattern.*

Given any $X \notin \mathbf{MB}_T$*, we have to "visit"* X_1, X_2, X_3, X_4 *or* X_5 *(Figure 1-6), and then some* P, C *or* S *before accessing* T*. Then, although there may exist many possible paths from* X *to* T*, each of them must contain some pattern(s) of the 1-5 as listed above; hence, we can infer that each of the possible path will be "blocked", and* X *is d-separated from* T*. Therefore, we conclude that* $\mathbf{MB}_T$ *d-separates* T *from all* $X \notin \mathbf{MB}_T$*.*

The proof that $\mathbf{MB}_T$ *is the minimal set is trivial by contradiction, and ignored here.* ∎

Therefore, d-separation actually bridges the semantic gap between the distribution and the graphical model, based on the faithfulness assumption. With d-separation, we are able to infer more conditional independence from the underlying DAG, in addition to the known Markov property.

2.6.3 MMPC/MB Algorithm

The overall Max-Min Markov blanket (MMMB, Figure 1-7) algorithm is composed of two steps. First, it discovers $\mathbf{PC}_T$ by MMPC(T). Then it attempts to identify $\mathbf{Sp}_T$. Any $X \in \mathbf{Sp}_T$ is known as the parent of some child(ren) of T, which suggests that they should belong to $\cup_{X \in \mathbf{PC}_T} \mathbf{PC}_X$, i.e. the union of the parents and children of the parents and children set. However, this union set, $\mathbf{MB}_T^G$(Line 3 of MMMB), also includes the children of the children of T, the parents of the parents of T, and the children of the parents of T. Thus, it is a superset of $\mathbf{MB}_T$, and those false positives need to be filtered out.

MMPC(T: Target, D: Dataset, ε: SignificanceValue)
{
//add candidate true positives to $\mathbf{PC}_T^G$
1. $\mathbf{PC}_T^G \leftarrow \{\ \}$;
2. **repeat**
3. **for**$(\forall X \in \mathbf{U} \backslash \mathbf{PC}_T^G \backslash \{T\})$**do**
4. $Sepset_{T,X} \leftarrow argmin_{\mathbf{Z} \subseteq \mathbf{PC}_T^G} I_D(T, X|\mathbf{Z})$;
5. $Y \leftarrow argmax_{X \in \mathbf{U} \backslash \mathbf{PC}_T^G \backslash \{T\}} (T, X|Sepset_{T,X})$;
6. **if**$(I_D(T, Y|Sepset_{T,Y}) \leq \varepsilon)$**then**
7. $\mathbf{PC}_T^G \leftarrow \mathbf{PC}_T^G \cup \{Y\}$;
8. **end if**
9. **until** $\mathbf{PC}_T^G$ does not change
//remove false positives from $\mathbf{PC}_T^G$
10. **for**$(\forall X \in \mathbf{PC}_T^G)$**do**
11. **if**$(I_D(T, X|\mathbf{Z}) > \varepsilon, for\ some\ \mathbf{Z} \subseteq \mathbf{PC}_T^G \backslash \{X\})$ **then**
12. $\mathbf{PC}_T^G \leftarrow \mathbf{PC}_T^G \backslash \{X\}$;
13. **end if**
14. **end for**
15. **return** $\mathbf{PC}_T^G$;
}

MMMB(T: Target, D: Dataset, ε: SignificanceValue)
{
// add true positives to MB
1. $\mathbf{PC}_T \leftarrow MMPC(T)$;
2. $\mathbf{MB}_T \leftarrow \mathbf{PC}_T$;
3. $\mathbf{MB}_T^G \leftarrow (\mathbf{PC}_T \cup_{\mathbf{PC}_T} MMPC(X)) \backslash \{T\}$;
// add more true positives to $\mathbf{MB}_T$
4. **for**$(\forall X \in \mathbf{MB}_T^G \backslash \mathbf{PC}_T)$ **do**
5. *find any* $\mathbf{Z}\ st. I_D(T, X|\mathbf{Z}) > \varepsilon\ and\ T, Y \notin \mathbf{Z}$;
6. **for**$(\forall Y \in \mathbf{PC}_T)$ **do**
7. **if**$(I_D(T, X|\mathbf{Z} \cup \{Y\} \leq \varepsilon)$**then**
8. $\mathbf{MB}_T \leftarrow \mathbf{MB}_T \cup \{X\}$;
9. **end if**
10. **end for**
11. **end for**
12. **return** $\mathbf{MB}_T$;
}

Figure 1-7: MMPC/MB algorithm

Given $X \in \mathbf{Sp}_T$ but not adjacent to T, it has the following property: conditioned on any subset that includes a common child (or children), X and T are dependent (**Theorem 1.4**, part 2). This property is not owned by the false positives in $\mathbf{MB}_T^G$, so it can be used to filter them out. One problem with checking the property directly is that we do not know which nodes in $\mathbf{PC}_T$ are actually children. Another problem is that it is inefficient to condition on all possible subsets.

Fortunately, MMMB overcomes both of these problems. First, it identifies a subset $Sepset_{T,X}$ that d-separates X from T (Line 5, MMPC), and caches it for later reference. Now if there is a variable $Y \in \mathbf{PC}_T$, such that $\sim I(X, T|\{Y\} \cup Sepset_{T,X})$, then Y has to be a child of T and X has to be a spouse of T. This is from the definition of the d-separation. The reverse also holds, so if there is no node Y for which the condition holds, X cannot be a spouse of T and it can be filtered out.

Tsamardinos et al. falsely proved in [21] that, under the assumptions of faithfulness and correct (in)dependence test, the output of MMPC is $\mathbf{PC}_T$ In practice, MMMB performs a test if it is reliable and skips it otherwise. MMMB follows the same criterion as IAMB and Fast-IAMB to decide whether a test is reliable or not. MMMB is data efficient because the number of instances required to identify $\mathbf{MB}_T$ does not depend on the size of $\mathbf{MB}_T$ but on the topology of $\mathbb{G}$. The experiments done in [21] shows that the algorithm is able to scale to problems with thousands of features, which actually reflects its merit of data efficiency.

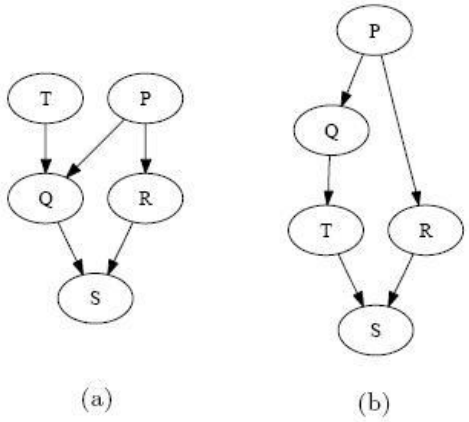


Figure 1-8: Two examples that MMPC/MB produces incorrect results

However, Pena et al. showed that MMPC does not guarantee to produce correct $\mathbf{PC}_T$ [13]. They pointed out that the flaw in the proof is the assumption that if $X \notin \mathbf{PC}_T$, then $I(X, T|\mathbf{Z})$ for some $\mathbf{Z} \sqsubseteq \mathbf{PC}_T$ and thus, any node not in $\mathbf{PC}_T$ that enters $\mathbf{PC}_T^C$ at line 7 is removed from it at line 12. This is not always true for the descendants of T, and it could be illustrated by running MMPC(T) with data faithful to the DAG(a) in Figure 1-8. Neither P nor R enters $\mathbf{PC}_T^C$ at line 7 because $I(T, P|\emptyset)$ and $I(T, R|\emptyset)$. Q enters $\mathbf{PC}_T^C$ because T is not independent for all $\mathbf{Z}$ such that $T, Q \notin \mathbf{Z}$. S enters $\mathbf{PC}_T^C$ because $\sim I(T, S|\emptyset)$ (since the path $T \rightarrow Q \leftarrow S$ is NOT blocked) and $\sim I(T, S|Q)$

(since the path $T \rightarrow Q \leftarrow P \rightarrow R \rightarrow S$ is NOT blocked). Then $\mathbf{PC}_T^C = \{Q, S\}$ at line 9. Neither Q nor S leaves $\mathbf{PC}_T^C$ at line 11. Consequently, the output of MMPC includes S which is not in $\mathbf{PC}_T$; therefore, MMPC does not guarantee the correct output under the faithfulness assumption. This example also illustrates that (1) partial $\mathbf{MB}_T$ can not completely shield T completely from outside variables; (2) if we have $\mathbf{MB}_T = \{Q, P\}$, then all the paths from T to $\{R, S\}$ will be blocked; (3) there is no subset of such $\mathbf{MB}_T$ to satisfy this condition.

Furthermore, Pena et al. showed that MMMB is not always true even if MMPC were correct under the faithfulness assumption. With DAG (b) in Figure 1-8, let us assume that MMPC is correct under the faithfulness assumption. Then, $\mathbf{MB}_T = \mathbf{PC}_T = \{Q, S\}$ and $\mathbf{MB}_T^C = \{P, Q, R, S\}$ at line 3 (of MMMB). P enters $\mathbf{MB}_T$ at line 8 if $\mathbf{Z} = \{Q\}$ at line 5, because (1)both $P \rightarrow Q \rightarrow T$ and $P \rightarrow R \rightarrow S \leftarrow T$ are blocked with instantiated Q, which means that $I(T, P|Q)$ (line 5); With $\mathbf{Z} = \{Q\}$, $\sim I(T, P|\{Q\} \cup \{S\})$ since the path $P \rightarrow R \rightarrow S \leftarrow T$ is NOT blocked with instantiated S (line 7). Consequently, the output of MMMB can include P which is not in $\mathbf{MB}_T$ and, thus, MMMB does not guarantee the correct output under the faithfulness assumption even if MMPC were correct under this assumption.

In [40], Tsamardinos et al. identify the flaw in MMPC and propose a corrected MMPC (CMMPC, Figure 1-9). The output of MMPC must be further processed in order to obtain $\mathbf{PC}_T$, because it may contain some descendants of T in G other than its children. Fortunately, these nodes can be easily identified: If X is in the output of MMPC(T), then X is a descendant of T in G other than one of its children iff T is in the output of MMPC(X). However, as shown above, correcting MMPC does not ensure MMMB is correct.

Different from IAMB and GS, which condition on the full candidate MB (CMB), MMPC or CMMPC tries all subsets of the candidate $\mathbf{PC}_T$ (CPC) or CMB in an attempt to d-separate all nodes not in the local neighborhood. Obviously, conditioning on the full CMB instead of all subsets of it significantly reduces the time complexity, but the sample requirements of the algorithms also increase exponentially. Therefore, MMPC/MB is the first valuable effort to improve the data efficiency of such category of algorithms, though it doesn't always produce correct outcome.

```
CMMPC(T: Target, D: Dataset, ε: SignificanceValue)
{
1.  PC_T^G = { };
2.  for(∀X ∈ MMPC(T)) do
3.      if(T ∈ MMPC(X)) then
4.          PC_T^G ← PC_T^G ∪ {X};
5.      end if
6.  end for
```

Figure 1-9: CMMC, Corrected MMPC

2.7 HITON-PC/MB

HITON [25](pronounced “hee-tón, it is from Greek, means “cover”, “cloak” or “blanket”) is also the work by the authors of IAMB, and can be viewed as an effort to further make the induction of Markov blanket more data efficient to meet the challenge in the biomedical field where sample sizes are typically limited (and often sample-to-variable ratios are very small). HITON also requires the same two assumptions as its ancestors IAMB: faithfulness and correct (in)dependence tests.

```
HITON − PC(T: Target, D: Dataset, ε: SignificanceValue)
{
1.  PC_T ← { };
2.  PC_T^G ← U\{T};
3.  repeat
        //add the best candidate to PC_T
4.      Y ← argmin_{X∈PC_T^G} I_D(T, X|{});
5.      PC_T ← PC_T ∪ {Y};
6.      PC_T^G ← PC_T^G\{Y};
        //remove false positives from PC_T^G
7.      for(∀X ∈ PC_T) do
8.          if(I_D(T, X|Z) > ε for some Z ⊆ PC_T \{X}) then
9.              PC_T ← PC_T\{X};
10.         end if
11.     end for
12. until PC_T^G ← {}
13. return PC_T;
}
```

```
HITON − MB(T: Target, D: Dataset, ε: SignificanceValue)
{
    // add true positives to MB
1.  PC_T ← HITON − PC(T);
2.  MB_T^G ← (PC_T ∪_{X∈PC_T} HITON − PC(X))\{T};
    // remove false positives from MB_T^G
3.  for(∀X ∈ MB_T^G) do
4.      for(∀Y ∈ PC_T) do
5.          if(I_D(T, X|Z ∪ {Y} > ε for some Z ⊆ (U\{T, X, Y})) then
6.              MB_T^G ← MB_T^G\{{X};
7.          end if
8.      end for
9.  end for
10. return MB_T^G
}
```

Figure 1-10: HITON-PC/MB algorithm

HITON works in a similar manner as MMMB. It first identifies the parents and child of T by calling HITON-PC and, then identifying the rest of the parents of the children of T in $\mathbb{G}$ via

HITON-MB (Figure 1-10). HITON-PC is similar to MMPC, with the exception that the former interleaves the addition of the nodes in $\mathbf{PC}_T^C$ to $\mathbf{PC}_T$(lines 4-6) and the removal from $\mathbf{PC}_T^C$ of the nodes that are not in $\mathbf{PC}_T$ but that have been added to $\mathbf{PC}_T^C$ by the heuristic at line 4 (lines 7-11). Note also that this heuristic is simpler than the one used by MMPC because the conditioning set is always the empty set. Aliferis et al. proved in [25] that, under the assumptions of faithfulness and correct independence test, the output of HITON-PC is $\mathbf{PC}_T$. However, this is not always true. The flaw in the proof is the same as that in the proof of correctness of MMPC. Running HITON-PC(T) with D faithful to the DAG (a) in Figure 1-8 can produce the same incorrect result as MMPC(T). Obviously, the flaw in HITON-PC can be fixed in the exactly the same way as the flaw in MMPC was fixed above.

Figure 1-10 outlines HITON-MB. The algorithm receives the target node T as input and returns $\mathbf{MB}_T$ in $\mathbf{MB}_T^C$ as output. HITON-MB is similar to MMMB. The algorithm works in two steps. First, $\mathbf{PC}_T$ and $\mathbf{MB}_T^C$ are initialized with $\mathbf{PC}_T$ and $(\mathbf{PC}_T \cup_{X \in \mathbf{PC}_T} \mathbf{PC}_X) \backslash \{T\}$ respectively, via the call of HITON-PC(T) (lines 1-2). Second, the nodes in $\mathbf{MB}_T^C$ that are neither in $\mathbf{PC}_T$ nor have a common child with T in G are removed from $\mathbf{MB}_T^C$ (Lines 3-9). This step is based on the following observation. If $X \in \mathbf{MB}_T^C$ and $Y \in \mathbf{PC}_T$, then X must be removed from $\mathbf{MB}_T^C$ iff $I(T, X|\mathbf{Z} \cup \{Y\})$ for some $\mathbf{Z}$ such that $T, X \notin \mathbf{Z}$. Aliferis et al. also prove that the output of HITON-MB is $\mathbf{MB}_T$ [25]. However, this is not always true even if HITON-PC were correct under the faithfulness assumption. The flaw in the proof is the observation that motivates the second step of HITON-MB, which is not true. This is illustrated by running HITON-MB(T) with D faithful to the DAG(b) in Figure 1-8. Let us assume that HITON-PC is correct under the faithfulness assumption. Then $\mathbf{PC}_T = \{Q, S\}$ and $\mathbf{MB}_T^C = \{P, Q, R, S\}$ at Line 3. P and R are removed from $\mathbf{MB}_T^C$ at line 6 because $Q \in \mathbf{PC}_T$, $I(T, R|Q)$ and $I(T, P|Q)$. Therefore, $\mathbf{MB}_T^C = \{Q, S\}$ at line 10. Consequently, the output of HITON-MB does NOT contain R, the spouse of T. Thus, HITON-MB does not guarantee the correct output even if HITON-PC were correct.

The experiments done in [25] show that the algorithm is able to scale to problems with thousands of features. Though it is not always correct, HITON-PC/MB still is recognized as another meaningful effort for an efficient learning algorithm of Markov blanket discovery without having to learn the whole Bayesian network.

2.8 PCMB

2.8.1 Motivation and Theoretical Foundation

Although neither MMPC/MB nor HITON-PC/MB is sound in theory, they represent a novel direction of learning Markov blanket in a more economic and practical manner, i.e. improving the efficiency of data usage by making use of the underlying topology information. This is considered as a great progress compared with all previous works, and it makes it possible for this kind of algorithm to work in many modern applications where high dimension is involved but collecting training data may be costly. Even for scenarios with relatively large volume of data, reducing the degrees of freedom of statistical tests may also increase the reliability of the results. Following this path, Pena & al who are the first ones to point out the flaw of MMPC/MB and HITON-PC/MB proposed a similar but sound algorithm, called PCMB (Parents and Children based Markov Blanket algorithm) [13]. It relies on the same two assumptions as required by MMPC/MB and HITON-PC/MB: faithfulness and correct statistical test. Similarly, PCMB induces MB via the recognition of direct connection, i.e. parents and children about any variable of interest, just like MMPC/MB and HITON-PC/MB do, which may explain where its name comes from.

Some background knowledge and theory about Bayesian network are covered in section 2.6.1. In this section, additional theorems necessary for the explanation and proof are presented for later reference, considering that our work is built on the same set of theoretical basis.

Theorem 1.6 Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$. Any probability distribution P satisfies the following four properties: (1) ***symmetry*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \Rightarrow I(\mathbf{Y}, \mathbf{X}|\mathbf{Z})$, (2) ***decomposition*** $I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$, (3) ***weak union*** $I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y}|\mathbf{Z} \cup \mathbf{W})$, and (4) ***contraction*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z} \cup \mathbf{W}) \wedge I(\mathbf{X}, \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z})$. If P is **strictly positive**, then P satisfies the previous four properties plus the ***intersection property*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z} \cup \mathbf{W}) \wedge I(\mathbf{X}, \mathbf{W}|\mathbf{Z} \cup \mathbf{Y}) \Rightarrow I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z})$. If P is **faithful** to a DAG $\mathbb{G}$, then P satisfies the previous five properties plus the ***composition property*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \wedge I(\mathbf{X}, \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z})$ and the ***local Markov property*** $I(X, \mathbf{ND}_X \backslash \mathbf{Pa}_X | \mathbf{Pa}_X)$, where $\mathbf{ND}_X$ denotes the non-descendants of X, and $\mathbf{Pa}_X$ for the parents of X [2, 28].

To make later references easier, we abstract those related properties of a probability distribution faithful to a DAG $\mathbb{G}$ as:

Corollary 1.1 Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$. Any probability distribution P faithful to a DAG $\mathbb{G}$ satisfies the following seven properties: (1) ***symmetry*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \Rightarrow I(\mathbf{Y}, \mathbf{X}|\mathbf{Z})$, (2) ***decomposition*** $I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$, (3) ***weak union*** $I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y}|\mathbf{Z} \cup \mathbf{W})$, and (4) ***contraction*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z} \cup \mathbf{W}) \wedge I(\mathbf{X}, \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z})$; (5) ***intersection*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z} \cup \mathbf{W}) \wedge I(\mathbf{X}, \mathbf{W}|\mathbf{Z} \cup \mathbf{Y}) \Rightarrow I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z})$; (6) ***composition*** $I(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \wedge I(\mathbf{X}, \mathbf{W}|\mathbf{Z}) \Rightarrow I(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z})$; (7) ***local Markov property*** $I(X, \mathbf{ND}_X \backslash \mathbf{Pa}_X | \mathbf{Pa}_X)$.

2.8.2 Algorithm Specification

PCMB identifies $\mathbf{PC}_T$ using the subroutines *GetPC*, and *GetPC* calls *GetPCD* to get the candidates. *GetPCD* receives the target node T as input and returns a superset of $\mathbf{PC}_T$ in $GetPCD.PCD$ (for easy reference, we attach the procedure name in front of the variable) as output. This superset contains false positives, nodes that do not belong in $\mathbf{PC}_T$. The algorithm tries to minimize the number of false positives, and it repeats three steps until $GetPCD.PCD$ does not change. First, some false positives are removed from $GetPCD.CanPCD$ (lines 4-11). This step is based on the observation that $X \in \mathbf{PC}_T$ iff $\sim I(T, X|\mathbf{Z})$ for all $\mathbf{Z}$ such that $T, X \notin \mathbf{Z}$. Second, the candidate most likely to be in $\mathbf{PC}_T$ is added to $GetPCD.PCD$ and removed from $GetPCD.CanPCD$ (lines 12-15). Since this step is based on the heuristic at line 13, some false positives may be added to PCD as well. Some of these nodes are removed from $GetPCD.PCD$ in the third step (lines 16-23). This step is based on the same observation as the first step. In $GetPCD$, the separator set corresponding to T and X (if there is, as found at Line 6 and 18) is cached and denoted with $Sepset_{T,X}$.

Theorem 1.7 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution P faithful to a DAG $\mathbb{G}$, *GetPCD*(T) returns a superset of $\mathbf{PC}_T$ that does not include any node in $\mathbf{ND}_T \backslash \mathbf{Pa}_T$ [13].

The output of *GetPCD* may still contain some descendants of T in $\mathbb{G}$ other than its children. These nodes can be easily identified. If X is in the output of *GetPCD*(T), then X is a descendant of T in $\mathbb{G}$ other than one of its children iff T is not in the output of *GetPCD*(X). *GetPC*, which is

outlined in Figure 1-11, implements this observation, and its correctness is proved by the author, see **Theorem 1.8**.

```
GetPCD(T: Target, D: Dataset, ε: SignificanceValue)
{
1.  PCD ← { };
2.  CanPCD ← U\{T};
3.  repeat
4.      //Remove false positives from CanPCD
5.      for(∀X ∈ CanPCD)do
6.          Sepset_{T,X} ← argmin_{Z⊆PCD} I_D(T, X|Z);
7.      for(∀X ∈ CanPCD)do
8.          if(I_D(T, X|Sepset_{T,X}) > ε) then
9.              CanPCD ← CanPCD\{X};
10.         end if
11.     end for
12.     //Add the best candidate to PCD
13.     Y ← argmax_{X∈CanPCD} I_D(T, X|Sepset_{T,X});
14.     PCD ← PCD ∪ {Y};[1-3]
15.     CanPCD ← CanPCD\{Y};
16.     //Remove false positives from PCD
17.     for(∀X ∈ PCD)do
18.         Sepset_{T,X} ← argmin_{Z⊆PCD} I_D(T, X|Z);
19.     for(∀X ∈ PCD)do
20.         if(I_D(T, X|Sepset_{T,X}) > ε) then
21.             PCD ← PCD\{X};
22.         end if
23.     end for
24. until PCD does not change;
25. return PCD;
}
```

```
GetPC(T: Target, D: Dataset, ε: SignificanceValue)
{
1.  PC_T^C ← { };
2.  for(∀X ∈ GetPCD(T))do
3.      if(T ∈ GetPCD(X))then
4.          PC_T^C ← PC_T^C ∪ {X};
5.  return PC_T^C;
}

PCMB(T: Target, D: Dataset, ε: SignificanceValue)
{
    // add true positives to MB
1.  PC_T ← GetPC(T);
2.  MB_T^C ← PC_T;
    // add more true positives to MB
3.  for(∀X ∈ PC_T) do
4.      for(∀Y ∈ GetPC(X)) do
5.          if(Y ∉ PC_T) then
6.              find Z st. I_D(T, Y|Z) > ε and T, Y ∉ Z;
7.              if(I_D(T, Y|Z ∪ {X}) ≤ ε)then
8.                  MB_T^C ← MB_T^C ∪ {Y};
9.              end if
10.         end if
11.     end for
12. end for
13. return MB;
}
```

Figure 1-11: PCMB Algorithm.

Theorem 1.8 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed (i.i.d.) sample from a probability distribution P faithful to a DAG $\mathbb{G}$, *GetPC*(X) returns $\mathbf{PC}_T$ [13].

PCMB receives the target node T as input and returns $\mathbf{MB}_T$ as output. The algorithm works in two steps. Firstly $\mathbf{MB}_T^C$ is initialized with $\mathbf{PC}_T$ by calling *GetPC* (line 2). Secondly, the parents of the children of T in $\mathbb{G}$ that are not yet in $\mathbf{MB}_T^C$ are added to it (lines 3-12). This step is base on the

following observation. The parents of the children of T in $\mathbb{G}$ that are missing from $\mathbf{MB}_T^C$ at line 3 are those that are non-adjacent to T in $\mathbb{G}$. Therefore, if $Y \in \mathbf{PC}_T$, $X \in \mathbf{PC}_Y$ and $X \notin \mathbf{PC}_T$ and, then X and T are non-adjacent parents of Y iff $\sim I(T, X|\mathbf{Z} \cup \{Y\})$ for any $\mathbf{Z}$ such that $I(T, X|\mathbf{Z})$ and $T, X \notin \mathbf{Z}$. **Note that Z can be efficiently obtained at line 6: *GetPCD* must have found such Z and have cached it with $Sepset_{T,X}$ as we mentioned above.**

Theorem 1.9 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution P faithful to a DAG $\mathbb{G}$, PCMB(T) returns $\mathbf{MB}_T$ [1].

In practice, PCMB follows the same criterion (equation (1.5) as IAMB, MMMB and HITON-MB to decide whether a test is reliable or not. PCMB is data efficient like MMMB and HITON-MB since the number of instances required to identify $\mathbf{MB}_T$ does not depend on the size of $\mathbf{MB}_T$ but on the topology of $\mathbb{G}$, but it is the first such kind of progress proved sound. Though one experiment in the original text [13] demonstrates that PCMB scales to one KDD Cup problem with thousands of features, it is shown as actually quite time inefficient by our empirical studies in Chapter 4, much slower than another algorithm we developed and that is introduced later, IPC-MB.

Chapitre 3 A NOVEL ALGORITHM FOR LOCAL LEARNING OF MARKOV BLAKNET : IPC-MB

3.1 Motivation

From the review in the previous two chapters, we can see a clear progress towards efficiently deriving an MB from data. Beginning with Koller and Sahami's work, it was shown that the Markov Blanket is the optimal feature subset, although the KS algorithm itself can't always guarantee correct output. Then, GS, IAMB and several variants were proposed. Compared with KS (algorithm), they are correct, simple and fast. However, they are known as very data inefficient [11, 13, 15, 21, 25, 26], which weakens their practical value, especially when the cost for collecting training data is high. MMPC/MB and HITON-PC/MB were therefore proposed aiming at reducing the critical requirement on the scale of training data. By putting the underlying topology information into consideration, MMPC/MB and HITON-PC/MB do proceed in the right direction to solve the data inefficiency problem, but unfortunately, neither of them is proved sound as they cannot always produce the correct outcome. Therefore, before this project starts, to the best of our knowledge, PCMB was the most promising algorithm that was published, known as sound, scalable and data efficient [26]. Even though Pena et al. proved that PCMB is correct, and showed that PCMB is scalable to large problems [13], there is still much space to improve based on our study, including accuracy, time and data efficiency.

3.1.1 Data efficiency, accuracy and time efficiency

The most common feature of these algorithms is that they are all based on conditional independence (CI) test. Based on our discussion in section (1.5, we can increase the reliability of CI tests by adding more observations, or reducing the degrees of freedom. Very often, we may have limited data; then, the only option is to reduce the number of variables as contained in the conditioning set of CI test (to make the discussion easier, we assume that each variable has the same number of values). In previous works [13, 21, 25], GS, IAMB and its variants were known to perform CI tests with conditioning set as large as $\mathbf{MB}_T$. In fact, considering that false positives (i.e. $X \in \mathbf{U} \backslash \mathbf{MB}_T$) may be added to the candidate MB container, $\mathbf{MB}_T^C$, in the growing phase,

the actual conditioning set can be even larger than $\mathbf{MB}_T$. For example, given an initially empty $\mathbf{MB}_T^C$ in the growing phase of IAMB, $X \in \mathbf{Des}_T$ will fail the CI test $I_D(X, T|\emptyset)$ (since at least we have an open path), and may be added into $\mathbf{MB}_T^C$ (Line 6 of IAMB, Figure 1-3). Upon the introduction of the first false positive, the door hence becomes open to more false ones, which possibly results in cascading errors due to the common way that independence-based algorithms work: their decision on which test to perform next typically depends on the outcomes of previous ones [37]. Thus, a single error in a statistical test, especially in the early stage, can be propagated by the subsequent choices of tests to be performed by the algorithm.

In IAMB, assuming that we have enough instances to allow the search to continue until no more can be added, we may have the $\mathbf{MB}_T^C$ set (much) larger than $\mathbf{MB}_T$ by the end of the growing phase as discussed in the last paragraph, instead of merely "as large as $\mathbf{MB}_T$ " as it was reported [13, 21]. If we have only limited observations, we may give up the search in the growing phase if there are not enough instances to support reliable statistical tests any more, with a $\mathbf{MB}_T^C$ set containing a subset of the target $\mathbf{MB}_T$ plus some false positives. These initial misclassified variables could impact the final accuracy (or recall, more specifically) because the shrinking phase can possibly help to filter out false positives as contained in $\mathbf{MB}_T^C$. Our experiments in Section 4.4 confirm that the actual performance as achieved by IAMB is not balanced, with precision level much higher than recall; additionally, its general accuracy performance is far below that of PCMB and our IPC-MB.

So, IAMB's poor accuracy performance is actually caused by its search strategy which doesn't make efficient use of observations. To improve the efficiency of data usage, PCMB makes use of the known topology information, and takes the divide-and-conquer strategy by first finding $\mathbf{PC}_T$, and then $\mathbf{Sp}_T$. In the inference of $\mathbf{PC}_T$, the authors of PCMB follow the first conclusion of **Theorem 1.4** by checking if $I(T, X|\mathbf{S})$ for each $\mathbf{S} \subseteq \mathbf{U}\backslash\{T, X\}$. To ensure correctness as well as to control the size of the conditioning set, PCMB interleaves shrinking and growing phases in GetPCD (Figure 1-11). Though this is effective to ensure sound outcome and efficient usage of data, it is time consuming since CI tests with all subsets of PCD or $CanPCD$ (PCD and $CanPCD$ are containers used to store found parents/children/descendants and candidate PCD respectively) have to be conducted for each iteration in GetPCD. Therefore, the accumulated time cost resulting from the many calls of GetPCD in PCMB will be considerable. As we will see in

Chapter 4, PCMB may be even more time consuming than the PC algorithm [14] which outputs the whole Bayesian network. This caveat is not reported in its original publication [13], and raises the issue of its actual scalability, at least from the perspective of time complexity.

In summary, we conjecture that although it is a valid algorithm for inducing Markov blankets, IAMB is data inefficient and may produce poor result given limited data, while PCMB can be costly in terms of time.

3.1.2 Assumptions and overview of our work

Although we have raised issues with the previous work, we acknowledge the efforts and contributions by predecessors since their findings illumined us all along:

- A Markov blanket is theoretically the optimal subset of variables for a classification task;
- IAMB is time efficient, but data inefficient since it may condition on the whole $\mathbf{MB}_T$ or an even larger set containing not only $\mathbf{MB}_T$ but some or all of $X \in \mathbf{U} \backslash \mathbf{MB}_T$. This is what we need to avoid if we want the solution practically valuable;
- Topology information may be of critical importance to avoid conditioning on needlessly large set [13, 21, 25].

In this project, we start by proposing a novel algorithm for learning MB, which minimizes the size of the conditioning set of CI tests during the search yielding better data efficiency than known algorithms. It is named Iterative Parent-Child based search of Markov Blanket (IPC-MB). Throughout our discussion below, we will assume the following assumptions:

- Faithfulness;
- No hidden variables;
- Reliable independence test (i.e. the test can tell us the (in)dependency if it holds in the distribution);
- Discrete observations;
- No missing values in observations.

Akin to PCMB, IPC-MB induces $\mathbf{MB}_T$ via the recognition of $\mathbf{PC}_T$ and $\mathbf{Sp}_T$, and follows these guiding principles:

- The induction of $\mathbf{MB}_T$ in IPC-MB proceeds in a manner of breadth-first search. It first determines the direct neighbors of T, i.e. $\mathbf{PC}_T$, and then the neighbors of each $Y \in \mathbf{PC}_T$, i.e. $\mathbf{PC}_Y$. This two-layer search permits us to not only find the true $\mathbf{PC}_T$ (how it is realized is discussed in Section 3.2.2 and 3.2.3), but prepare the search for spouse candidates considering that spouses must belong to some $\mathbf{PC}_Y$ where $Y \in \mathbf{PC}_T$ (more detail can be found in Section 3.2.3). Some additional checking is further applied to induce those true spouses from among the candidates (refer to Section 3.2.4 for more discussion). Hence, both the learning of $\mathbf{PC}_T$ and $\mathbf{Sp}_T$ depend, directly or indirectly, on the search of local neighbours, which explains the origin of our algorithm;
- In the induction of $\mathbf{PC}_X$ (here X can represent T and T's neighbours found in IPC-MB), we start with the assumption that all $\mathbf{U}\backslash\{X\}$ are X's parents or children. Then, it proceeds by checking and removing false positives, i.e. those actually belonging to $\mathbf{U}\backslash\mathbf{PC}_X$. Considering that (1) we are able to delete the link $X - Y$ if there exists a single positive CI test, with some $\mathbf{Z} \subseteq \mathbf{U}\backslash\{X, Y\}$ as the conditioning set, indicating that X and Y are independent; (2) the real network normally is not dense in connectivity and $\mathbf{PC}_X$ is small relative to $\mathbf{U}$ in most cases, then the removal of false positives is believed an effective approach to decrease the search space quickly. By removing those false positives, all or most of the remaining ones are expected to belong to $\mathbf{PC}_X$;
- During the process of filtering out $Y \in \mathbf{U}\backslash\mathbf{PC}_X$, the conditioning set in $I_D(X, Y|\mathbf{Z})$ starts with empty set, and grows one at time. Whenever Y is tested as conditionally independent with X given some $\mathbf{Z}$, it is considered as not belonging to $\mathbf{PC}_X$ and removed from $\mathbf{PC}_X^C$ (Candidate $\mathbf{PC}_X$) right now. Therefore, any decision on false positive is made with as small conditioning set as possible, which maximizes data efficiency;
- Meanwhile, since we start with the empty conditioning set, and each Y in the $\mathbf{PC}_X^C$ is tested given the current conditioning set(s), as many false positives are removed, and at an as early time as possible, which maximizes time efficiency;
- Therefore, IPC-MB is expected to solve the most severe shortcoming of IAMB and PCMB, thereby maximizing computational efficiency.

The rest of this chapter is organized as follows. The specification and proof of IPC-MB is covered in section 3.2 and 3.3. Complexity analysis is discussed in section 3.4, followed by the discussion of data efficiency and reliability in section 3.5. Analysis of our algorithm given polytree, one special type of Bayesian network, is presented in section 3.6. Section 3.7 discusses the parallel version of IPC-MB, and the final section provides some concluding remarks.

3.2 IPC-MB algorithm specification and proof

3.2.1 Overall description

A novel algorithm for learning Markov blanket is proposed by us in this section, based on a series of CI tests. Since it induces the target Markov blanket via iterative learning of parents and children, it is named as IPC-MB (Iterative Parent-Child based learning of Markov Blanket).

Although IPC-MB can be grouped into the category of constraint-based learning like HITON-PC/MB, MMPC/MB and PCMB, it differs from those three in the search of local neighbors of some variable X (i.e. $\mathbf{PC}_X$): IPC-MB initially assumes that all $\mathbf{U}\backslash\{X\}$ are connected (or adjacent) to X, and it tries to remove $Y \in \mathbf{U}\backslash\mathbf{PC}_X$ with $\mathbf{PC}_X$ left; however, the other three work to determine directly if $Y \in \mathbf{PC}_X$. For easy reference, we use $\mathbf{PC}_X^C$ to denote the candidate adjacent neighbors of X, and it is initialized as $\mathbf{U}\backslash\{X\}$. To realize that, IPC-MB starts with empty conditioning set ϕ, and removes from $\mathbf{PC}_X^C$ all Y that are known as conditionally independent from X by CI test $I_D(X, Y|\mathbf{Z})$ where $\mathbf{Z} = \emptyset$. Then, the allowed conditioning set size grows by one, and the removing continues if there is $Y \in \mathbf{PC}_X^C$ which is known as independent from X conditioned on some $\mathbf{Z}$, where $\mathbf{Z} \subseteq \mathbf{PC}_X^C$ and $|\mathbf{Z}| = 1$. The search continues on in this way, with the conditioning set growing by one each time, and terminates when there are no CI tests remaining in $Y \in \mathbf{PC}_X^C$ (which will be discussed in 3.2.2). In so doing, false positives are removed by the lowest-order tests, resulting in a decreased search space. More importantly, minimizing the high-order tests reduces the risk of non-reliable tests, while improving the overall reliability of the algorithm especially when the sample size is limited. This is important since learning built on statistical tests suffers most from the curse of dimensionality [38]. However, the other methods (HITON-PC/MB, MMPC/MB and PCMB) have to know if X is conditionally dependent from $Y \in \mathbf{U}\backslash\{X\}$ given all $\mathbf{Z} \subseteq \mathbf{U}\backslash\{X, Y\}$ before including it into $\mathbf{PC}_X$.

Similar to MMPC/MB, HITON-PC/MB and PCMB, the whole procedure of IPC-MB can be divided into two phases:

1. Firstly, it attempts to identify nodes directly connected to T among $\mathbf{U}$, i.e. $\mathbf{PC}_T$. This actually is achieved by two steps, recognizing the candidate parents and children, followed by filtering out false positives (if there are any) to reach the true $\mathbf{PC}_T$;
2. Secondly, it induces the $\mathbf{Sp}_T$ from the candidates prepared in the first phase. Note that for the $X \in \mathbf{PC}_T \cap \mathbf{Sp}_T$, they are recognized as $\mathbf{PC}_T$ first and be included into $\mathbf{MB}_T$ in the first step.

The first phase will be discussed with detail in section 3.2.2 and 3.2.3, and the second phase with section 3.2.4.

3.2.2 Learn Parent/Child Candidates

The discovery of parent/child is critical to the efficiency of the local search approach of this algorithm of IPC-MB. Given a variable X, the *FindCanPC* procedure (Figure 3-1) aims to identify the target's parents and children, though, but descendants may be output as well. *FindCanPC* has four input parameters:

1. T, the target variable;
2. $Y \in \mathbf{PC}_T^C$, the candidate parents and children initialized as $\mathbf{U}\backslash\{T\}$. Then, obviously, $\mathbf{PC}_T \sqsubseteq \mathbf{PC}_T^C$;
3. D, the dataset used for learning;
4. ε, significance threshold value used in determining if a CI test indicates positive CI relationship (when the result of test is larger than ε) or not (when the result is smaller or equal to ε), i.e. significant or not. Empirical choice may be 0.01 or 0.05. *Note: As mentioned in section 2.2, we implement G^2 test and apply it for all algorithms covered in our discussion.*

```
FindCanPC(T: Target, PC_T^C: Adjacency set, D: Dataset, ε: SignificanceValue)
{
1.  NonPC ← ∅;
2.  cutSetSize ← 0;
3.  repeat
4.    for( ∀X ∈ PC_T^C) do
5.      for(S ⊆ PC_T^C\{X}) with |S| = cutSetSize) do
6.        if(I_D(T, X|S) ≤ ε) then
7.          NonPC ← NonPC ∪ {X};
8.          Sepset_{T,X} ← S;        //Cache for later reference
9.          break;
10.       end if
11.     end for
12.   end for
13.   PC_{T_T}^C ← PC_T^C\NonPC;
14.   NonPC ← ∅;
15.   cutSetSize++;
16. until (|PC_T^C| ≤ cutSetSize)
17. return PC_T^C;
}
```

Figure 3-1: *FindCanPC* algorithm and its pseudo code.

FindCanPC begins with the assumption that T is dependent over all $X \in \mathbf{PC}_T^C$, which means that T is connected with each $X \in \mathbf{PC}_T^C$ given the faithfulness assumption. Then, it tries to determine whether or not each such edge $T - X$ should be deleted, which corresponds to removing false positives from $X \in \mathbf{PC}_T^C$. This is achieved by three embedded loops (*Note that we assume there are enough observations for learning here, i.e. the discussion over the reliability of the CI tests is postponed to section 3.5*):

1. *Repeat…until* (the outmost loop) (Line 3 – 16). It starts with empty conditioning set ($cutSetSize = 0$), and exits when that $|X \in \mathbf{PC}_T^C|$ is equal to $cutSetSize$. In addition to the two embedded inner *for…do…* loops, we find additional instructions (line 13-15):

   a) If there are false positives found, i.e. $NonPC \neq \emptyset$, they are removed from $\mathbf{PC}_T^C$ by the end of this iteration. Hence, in the next iteration, we may have a smaller search space. If there are false positives removed in each iteration, the search space will continue to shrink;

b) Otherwise if $NonPC = \emptyset$, i.e. no false positive is found, nothing is done except for increasing the $cutSetSize$ by one.

2. *for(each* $X \in \mathbf{PC}_T^C$*) do…* (the middle layer) loop (Line 4-12). Upon entering this loop, each $X \in \mathbf{PC}_T^C$ is assumed to be connected with T. With conditioning sets of size $cutSetSize$, each $X \in \mathbf{PC}_T^C$ will be checked if it is conditionally independent with T (how this is done is discussed in the next point). If it is, X will be put into $NonPC$ and be removed from $\mathbf{PC}_T^C$ by the end of *do…while…* loop as we discussed above.

3. *for(each* $\mathbf{S} \subseteq \mathbf{PC}_T^C \backslash \{X\}$*) do…*loop (the inner one) (Line 5 – 11). With each $X \in \mathbf{PC}_T^C$ and given $cutSetSize$, X is checked if it is independent with T conditioned on some $\mathbf{S} \subseteq \mathbf{PC}_T^C \backslash \{X\}$, as tested by the statistical function I_D (line 6). ***Note that the number of*** I_D ***involved in the search is a critical measure that reflects the time complexity of this kind of algorithm, and we will discuss this topic in more details in 3.4.1 and 3.4.2.*** Anytime X is tested as independent with T, conditioned on some **S**, it is added to $NonPC$ (line 7) and exits from the current loop (line 9), ***which is the advantage of filtering false positives from an initial candidate set since we stop after a single negative CI test each time and we can start with the smallest conditioning set, instead of working from an empty candidate set where we would need to run all possible CI tests each time.*** For each rejected candidate, the found conditioning set **S** is denoted as $Sepset_{T,X}$, and cached for later reference (spouse learning in 3.2.4).

To better explain the algorithm, we illustrate the procedure by a simple example, given target T and $\mathbf{PC}_T^C = \{U, V, W, X, Y, Z\}$ initially:

1. $cutSetSize = 0$. *The following CI tests will be conducted:* $I_D(T, U|\emptyset)$, $I_D(T, V|\emptyset)$, $I_D(T, W|\emptyset)$, $I_D(T, X|\emptyset)$, $I_D(T, Y|\emptyset)$ *and* $I_D(T, Z|\emptyset)$. *Assuming that only two nodes have a positive CI test,* $I_D(T, U|\emptyset) \leq \varepsilon$ *and* $I_D(T, V|\emptyset) \leq \varepsilon$, *then U and V are put into* $NonPC$ *at line 7; meanwhile,* $Sepset_{T,U} = \emptyset$ *and* $Sepset_{T,V} = \emptyset$ *are cached for later reference (line 8). At line 13, both U and V are removed from* $\mathbf{PC}_T^C$, *with updated* $\mathbf{PC}_T^C = \{W, X, Y, Z\}$. *With new* $cutSetSize=1$ *(by increasing with 1 at line 14), it is still smaller than* $|\mathbf{PC}_T^C|$ *(=4, line 16), we continue with the processing;*

2. $cutSetSize = 1$. *At most, the following groups of CI tests will be done:*

a) $I_D(T, W|X)$, $I_D(T, W|Y)$, $I_D(T, W|Z)$;

b) $I_D(T, X|W)$, $I_D(T, X|Y)$, $I_D(T, X|Z)$;

c) $I_D(T, Y|W)$, $I_D(T, Y|X)$, $I_D(T, Y|Z)$;

d) $I_D(T, Z|W)$, $I_D(T, Z|X)$, $I_D(T, Z|Y)$.

Each group above is about T and some $X \in \mathbf{PC}_T^C$ *conditioning on some* $\mathbf{S} \subseteq \mathbf{PC}_T^C \backslash \{X\}$ *and* $|\mathbf{S}| = 1$. *At any time, if we determine that T and X are conditionally independent (i.e.* $I_D(T, X|\mathbf{S}) \leq \varepsilon$*), the remaining CI tests in the corresponding group are ignored. For example, if* $I_D(T, W|X) \leq \varepsilon$, W *is added into* $NonPC$, *neither* $I_D(T, W|Y)$ *nor* $I_D(T, W|Z)$ *is necessary. That is why we mention "at most" above. Assuming* $I_D(T, W|X) \leq \varepsilon$ *and* $I_D(T, Y|X) \leq \varepsilon$, *we have updated* $\mathbf{PC}_T^C = \{Y, Z\}$ *and* $cutSetSize = 2$;

3. *Because* $|\mathbf{PC}_T^C| = cutSetSize = 2$, *we exit from the loop, with* $\mathbf{PC}_T^C = \{Y, Z\}$ *being returned. We can do so because with* $\mathbf{PC}_T^C = \{Y, Z\}$, *all possibly constructed CI tests, like* $I_D(T, Y|\emptyset)$, $I_D(T, Y|Z)$, $I_D(T, Z|\emptyset)$, *and* $I_D(T, Z|Y)$, *are conducted in previous iterations. Hence, there is nothing to do but exit.* ∎

Theorem 3.1 Under the assumptions that the independence tests are correct and that the learning data D is an i.i.d. sample from a probability distribution P faithful to a DAG G, given $\mathbf{PC}_T^C = \mathbf{U}\backslash\{T\}$, *FindCanPC* enables us to find the superset of $\mathbf{PC}_T$, denoted as $\mathbf{PC}_T^C$ (Candidate Parents and Children), and it has two properties: (1) for each $X \in \mathbf{PC}_T$, $X \in \mathbf{PC}_T^C$; and (2) there are some false positives contained in $\mathbf{PC}_T^C, \mathbf{PC}_T \subseteq \mathbf{PC}_T^C$.

Proof. *We need to prove the two properties respectively. The first one is proved by contradiction. With* $\mathbf{PC}_T^C = \mathbf{U}\backslash\{T\}$ *initially, it is assumed that there is some* $X \in \mathbf{PC}_T$ *not output by FindCanPC. Given the faithfulness assumption, it is known that if* $X \in \mathbf{PC}_T$, X *is connected to T directly. According to* **Theorem 1.4**, *for such* X , *it should NOT be independent of T given any conditioning set, i.e. should pass all* $I_D(T, X|\mathbf{Z})$ *as met in FindCanPC. Given correct statistic test, X would not be output by FindCanPC only when it fails on some* $I_D(T, X|\mathbf{S})$, *which is obviously contradictory with the fact that* $X \in \mathbf{PC}_T$. *Therefore, all* $X \in \mathbf{PC}_T$ *would be returned by FindCanPC.*

We prove the second property ($\mathbf{PC}_T \sqsubseteq \mathbf{PC}_T^C$) with the example of Figure 1-8(a) that FindCanPC may output some of T's descendants given some topology. With T and $\mathbf{PC}_T^C = \{Q, P, R, S\}$, it starts by connecting T with Q, P, R and S. With $cutSetSize = 0$, (1) both $T \to Q \leftarrow P$ and $T \to Q \to S \leftarrow R \leftarrow P$ are blocked due to the converging pattern at Q and S, and neither is instantiated, which implies that $I(T, P|\emptyset)$ since they are the only two paths between T and P; (2) both $T \to Q \leftarrow P \to R$ and $T \to Q \to S \leftarrow R$ are blocked due to the same reason, and we have $I(T, R|\emptyset)$. Hence, $NonPC = \{P, R\}$, $Sepset_{T,P} = \emptyset$ and $Sepset_{T,R} = \emptyset$. Then, with $cutSetSize = 1$ and $\mathbf{PC}_T^C = \{Q, S\}$, (1) $\sim I(T, Q|S)$ is trivial; (2) $\sim I(T, S|Q)$ since although $T \to Q \to S$ is blocked with Q instantiated, $T \to Q \leftarrow P \to R \to S$ is not blocked (P or R is needed, but both are absent). Therefore, no additional false positive is found, the search terminates with $\{Q, S\}$ being returned. Obviously, S is a false positive that is not filtered out. There are two possible paths from T to S: $T \to Q \to S$ and $T \to Q \leftarrow P \to R \to S$. Based on the d-separation concept, the minimum cut set to "block" T and S is $\{Q, R\}$ or $\{Q, P\}$. However, here, R and P have been deleted from $\mathbf{PC}_T^C$ when $cutSetSize = 0$, which prevents us from filtering S. Therefore, FindCanPC may output false positives, and $\mathbf{PC}_T \subseteq \mathbf{PC}_T^C$. Note that S is a false positive descendant of T based on **Definition 1.7**, *and as we will discuss right below, FindCanPC(X) may output X's descendants.* ■

Before we discuss how to filter out false positives as contained in the output of *FindCanPC*, it is necessary to study more closely how they occur.

Lemma 3.1 Given T and $\mathbf{PC}_T^C = \mathbf{U} \backslash \{T\}$, the output of *FindCanPC* will NOT contain non-descendants of T excluding T's parents, i.e. $\mathbf{ND}_T \backslash \mathbf{Pa}_T$.

Proof. *(1) The local Markov property (**Theorem 1.6**) tells us that T is independent of its non-descendants given the value of its parents, i.e. $I(T, \mathbf{ND}_T \backslash \mathbf{Pa}_T | \mathbf{Pa}_T)$; (2) It is known that $\mathbf{Pa}_T$ will always stay in $\mathbf{PC}_T^C$ (**Theorem 3.1**), i.e. $|\mathbf{Pa}_T| \leq |\mathbf{PC}_T^C|$; (3) The conditioning set starts with ϕ, so we are guaranteed to have a chance to condition $\mathbf{Pa}_T$ at $cutSetSize = |\mathbf{Pa}_T|$; (4) We check each $X \in \mathbf{PC}_T^C$ in each iteration, including the iteration of $cutSetSize = |\mathbf{Pa}_T|$. Therefore, each $X \in \mathbf{ND}_T \backslash \mathbf{Pa}_T$ is able to be successfully recognized given the test $I_D(T, X|\mathbf{Pa}_T)(\leq \varepsilon)$ and is filtered out as expected.* ■

Lemma 3.2 Given T and $\mathbf{PC}_T^C = \mathbf{U}\backslash\{T\}$, *FindCanPC* may or may not output descendants of T, and it depends on the underlying topology.

Proof. *(1) In* **Theorem 3.1**, *one example has been given to show that descendants of T may be missed from deletion. (2) Here, we give another example to show that descendants of T may not be output by FindCanPC. In Figure 3-2, if there are no dotted links $ND_1 - Des$ (between Non-Descendant and Descendant, and the direction of the edge does not matter) and $P - Des$ (between Parent and Descendant) in addition to $C \rightarrow Des$, it is trivial to know that Des is d-separated from T given C, so it will not be output by FindCanPC. Even with $ND_1 - Des$ and $P - Des$ added, we can still prove that the paths of $Des - ND_1 - P \rightarrow T$ and $Des \leftarrow P \rightarrow T$ are blocked by $\{P\}$ due to the existing of serial and diverging patterns respectively. Since P and C will always be output by FindCanPC, Des in this example then will never be output by FindCanPC.* ∎

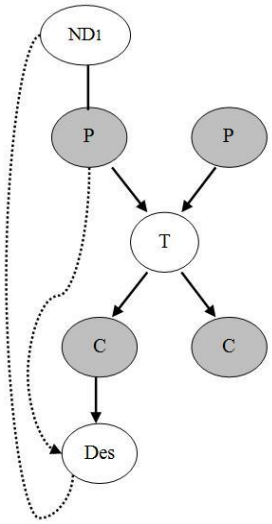


Figure 3-2: Possible connections between Non-Descendants/Parents/Children and descendant.

Theorem 3.2 Given T and $\mathbf{PC}_T^C = \mathbf{U}\backslash\{T\}$, *FindCanPC* may return false positives, and they may only be T's descendants, but not its non-descendants (excluding $\mathbf{Pa}_T$).

Proof. (1) $\mathbf{PC}_T^C = \mathbf{U}\backslash\{T\} = \mathbf{ND}_T \cup \mathbf{Des}_T$; (2) $\mathbf{ND}_T \cap \mathbf{Des}_T = \emptyset$. *These two facts, plus the proof of* **Lemma 3.1** *and* **Lemma 3.2**, *are enough to declare that non-descendants won't be output by FindCanPC, and some descendants may be output under some conditions though it is not expected.* ∎

In this section, we introduced the core module of IPC-MB, *FindCanPC*. We start with heuristics, followed by algorithm specification, and ended with a proof that *FindCanPC* will output all parents and children of T; additionally, we noted that some of T's descendants may also be falsely output depending on the underlying topology. In the next section, we will discuss how to construct a true parent-children set of T, i.e. $\mathbf{PC}_T$, by filtering those false positives that *FindCanPC(T)* may output.

3.2.3 Learn Parents/Children

As we discussed above, *FindCanPC(T)* itself does not guarantee to return exactly the parents and children of T (**Theorem 3.1**), but some descendants of T (**Theorem 3.2**). Unfortunately, candidate parents and children are mixed together, and therefore denoted as $\mathbf{PC}_T^C$ (Line 2 of *IPC-MB*, Figure 3-3). The container reserved for true Parents/Children, denoted with $\mathbf{PC}_T$, is initialized as empty (Line 3, *IPC-MB*).

Lemma 3.3 With $\mathbf{PC}_T^C \leftarrow$ *FindCanPC*(T), given each $X \in \mathbf{PC}_T^C$ and $\mathbf{PC}_X^C$ = *FindCanPC*(X), (1) if $T \in \mathbf{PC}_X^C$, X is known as a true parent/child, and should be added into $\mathbf{PC}_T$ (Line 7-10, IPC-MB); (2) if $T \notin \mathbf{PC}_X^C$, X is known as a false parent/child, and would be ignored with no further action.

Proof. ***Theorem 3.2*** *tells us that FindCanPC(T) may contain two types of output: true parents/children of T as expected, and descendants of T which are not desired. The proof contains two parts based on extra checking on each* $X \in \mathbf{PC}_T^C$*: what true is still recognized as true, but what false can be successfully filtered out.*

First, if $X \in \mathbf{PC}_T$, *obviously,* $T \in \mathbf{PC}_X$ *and T would be returned by FindCanPC(X) given* ***Theorem 3.1****. Then, our inference that "if* $X \in \mathbf{PC}_T^C$ *and* $T \in \mathbf{PC}_X^C$*, then the decision that X is true parent/child " is known correct.*

Second, we need to prove that if $X \in \mathbf{PC}_T^C$ *but* $T \notin \mathbf{PC}_X^C$, *then X is false parent/child. From* ***Theorem 3.2****, it is known that what false positives possibly output by FindCanPC*(T) *and FindCanPC(X) can only be T's* and *X 's descendants respectively. Assuming that* $X \in \mathbf{PC}_T^C$ but $X \notin \mathbf{PC}_T$, *can T be returned by FindCanPC*(X)*?*

This may happen only when T is its own descendant's descendant. Obviously, it is impossible since one cycle will happen. So, if $X \in \mathbf{PC}_T^C$ *and* $T \notin \mathbf{PC}_X^C$*, it can be inferred that X is NOT true parent/child, and should NOT be added into* $\mathbf{PC}_T$*.* ■

```
IPC − MB(T: Target, D: Dataset, ε: SignificanceValue)
{
    // Recognize T's parents and children
1.  PC_T^C ← U\{T};      // Candidate adjacency set
2.  PC_T^C ← FindCanPC(T, PC_T^C, D, ε);    //Candidate parents/children
3.  PC_T ← {};
4.  for(∀X ∈ PC_T^C) do
5.      PC_X^C ← U\{X};
6.      PC_X^C ← FindCanPC(X, PC_X^C, D, ε);
7.      if(T ∈ PC_X^C)then
8.          PC_T ← PC_T ∪ {X};      // One true parent/child is found
9.          Sp_{T,X}^C ← PC_X^C;      //Cache for future reference
10.     end if
11. end for
12. MB_T ← PC_T;
    // Recognize T's spouses
13. for(∀X ∈ PC_T) do
14.     for(∀Y ∈ Sp_{T,X}^C and Y ∉ MB_T) do
15.         if(I_D(T, Y|Sepset_{T,Y} ∪ {X}) > ε) then
16.             MB_T ← MB_T ∪ {Y};    // A true spouse is found
17.         end if
18.     end for
19. end for
20. return MB_T;
}
```

Figure 3-3: IPC-MB algorithm and its pseudo code.

Figure 3-4 demonstrates the effect of repeating the call of *FindCanPC* in IPC-MB. Although some false positives may be output by *FindCanPC(T)*, it is known that they can be successfully recognized and deleted in IPC-MB, as shown in **Lemma 3.3**. By repeating the call of *FindCanPC* for each $X \in \mathbf{PC}_T^C$, then we know that the results of $\mathbf{PC}_T$ at line 12 (IPC-MB) are exactly parents and children of T.

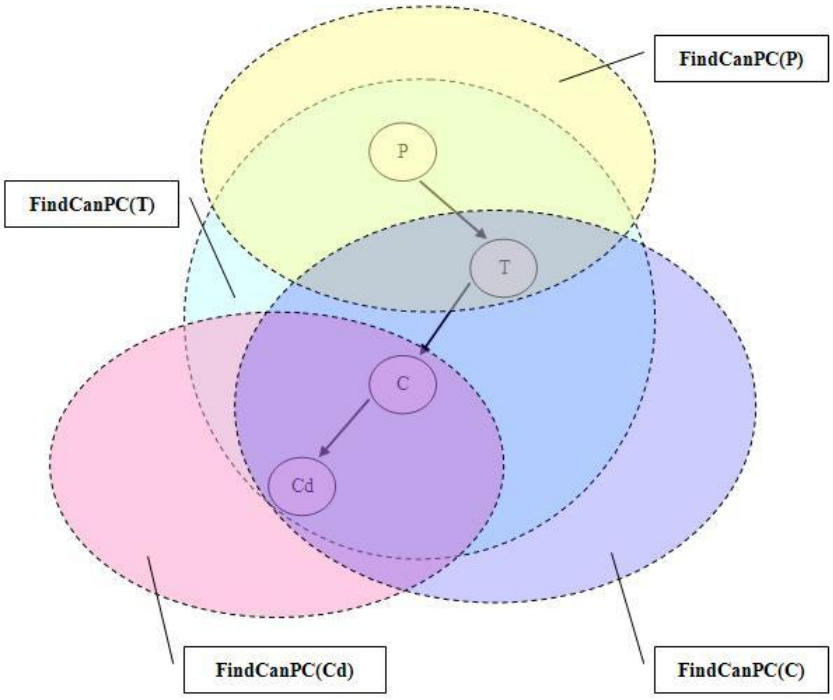


Figure 3-4: $\mathbf{PC}_T^C$ as output by *FindCanPC(T)*, and the output of typical $X \in \mathbf{PC}_T^C$, i.e. $\mathbf{PC}_X^C$.

Theorem 3.3 Under the assumptions that the independence tests are correct and that the learning data D is i.i.d sample from a probability distribution P faithful to a DAG G, IPC-MB allows us to find the complete and correct parents and children of one target, i.e. $\mathbf{PC}_T$ for T of interest.

Proof. (1) *Given* ***Theorem 3.1****,it is known that* $\mathbf{PC}_T^C$ *contains not only* $\mathbf{PC}_T$ *but, probably, some false positives; (2)****Theorem 3.2*** *tells us that those possible false positives can only be descendants of* T*; (3) With* ***Lemma 3.3****, we know that if* $X \in \mathbf{PC}_T^C$ *is a false positive descendant, it will be recognized. (4) Finally, since we apply the same verification procedure for each* $X \in \mathbf{PC}_T^C$*, we are guaranteed to have* $\mathbf{PC}_T$ *with all false positives being removed from* $\mathbf{PC}_T^C$. ■

By now, we have explained and proved that IPC-MB allows us to learn the complete parents and children of T of interest, i.e. $\mathbf{PC}_T$ (**Theorem 3.3**). It is noticed that the learning is built on a series of *FindCanPC(X)* , which exactly explains why our algorithm is called Iterative Parent-Child based learning of Markov Blanket (IPC-MB).

What left is the learning of T's spouses, i.e. $\mathbf{Sp}_T$. How to recognize $\mathbf{Sp}_T$ is discussed in the next section, but it is necessary to predict that it is built on the finding of *FindCanPC(X)* as well.

3.2.4 Learn Spouses

By the Line 12 of IPC-MB (Figure 3-3), we have $\mathbf{MB}_T = \mathbf{PC}_T$ as discussed in last section. In fact, we also have collected all candidate spouses of T with the repeated calls of *FindCanPC(X).*

Lemma 3.4 Given $X \in \mathbf{PC}_T^C$, if $T \in \mathbf{PC}_X^C$, $\mathbf{PC}_X^C$ contains candidate spouses of T if there are.

Proof. ***Theorem 3.1*** *tells us that* $\mathbf{PC}_X^C$*, the output of FindCanPC(X), contains all parents/children of* X*. Given* $X \in \mathbf{PC}_T^C$, if $T \in \mathbf{PC}_X^C$*, then* X *is known as a true parent/child (***Lemma 3.3***). If* X *is a child of* T*, and if it is a common child of* T *and some* Y*,* Y *must be returned by FindCanPC(X) . This applies to all* X*'s parents which are* T*'s spouses meanwhile.* ∎

All outputs of *FindCanPC*(X) regarding to such $X \in \mathbf{PC}_T^C$ are cached as $\mathbf{Sp}_{T,X}^C$ (Line 9, IPC-MB) with subscript (T, X) for later reference. Obviously, it contains more than what we want:

- T, since $T \in \mathbf{PC}_X^C$;
- True parents and/or children of T, which would be ignored;
- True spouses of T, i.e. those having X as their child as T. These are what we are interested to distinguished here;
- False positives (neither parents, children nor spouses of T).

Lemma 3.5 Given $\mathbf{PC}_X^C$ *=FindCanPC*(X), $\mathbf{Sp}_T \subset (\cup\, \mathbf{PC}_X^C)$, where $X \in \mathbf{PC}_T^C$ and $T \in \mathbf{PC}_X^C$, i.e. $X \in \mathbf{PC}_T$.

Proof. *Assume there exists some spouse* S *of* T *which is not contained in* $\cup_{X \in \mathbf{PC}_T} \mathbf{PC}_X^C$*, which means that* S *is not contained in any* $\mathbf{PC}_X^C$*, where* $X \in \mathbf{PC}_T^C$ *and* $T \in \mathbf{PC}_X^C$ *. This may happen only when (1) The common child of this* S *and* T *is not contained in* $\mathbf{PC}_T^C$, or (2) S *is not returned by its common child with* T*,* X*, though* $X \in \mathbf{PC}_T^C$*. Both cases are contradictory to the facts that FindCanPC(X) returns all parents and children of X (***Theorem 3.1***).* ∎

With **Lemma 3.5**, it is known that $\cup_{X \in \mathbf{PC}_T} \mathbf{PC}_X^C$ contain all candidate spouses of T, by Line 12 of *IPC-M*B, and it is denoted with shorthand $\mathbf{Sp}_T^C$. However, there are many false positives are known as contained in $\mathbf{Sp}_T^C$ as well, waiting for further processing.

Similarly to the discovery of parents and children of T, i.e. $\mathbf{PC}_T$, we depend on the underlying connectivity information to recognize $\mathbf{Sp}_T$ from $\mathbf{Sp}_T^C$. For any $X \in \mathbf{Sp}_T$, there are two facts

available for reference: (1) it has to belong to $\mathbf{PC}_Y$ and $Y \in \mathbf{PC}_T$; (2) it is independent of T as conditioned on $Sepset_{T,X}$ or $Sepset_{X,T}$ (that is why it is not included in $\mathbf{PC}_T$) , but it is dependent with T conditioned on $Sepset_{T,X} \cup \{Y\}$ or $Sepset_{X,T} \cup \{Y\}$. The first observation is obvious given the underlying topology, and the second is based on **Theorem 1.4**.

Lemma 3.6 Given each $X \in \mathbf{Sp}_T^C$ but $X \notin \mathbf{PC}_T$, there must exist some $\mathbf{Z}$, $\emptyset \subseteq \mathbf{Z} \subset \mathbf{U}$, such that $I(T, X|\mathbf{Z})$.

Proof. *The proof is trivial since if there is no such* $\mathbf{Z}$*,* X *should be in* $\mathbf{PC}_T$.

Lemma 3.7 In IPC-MB, for each $X \in \mathbf{Sp}_T^C$ but $X \notin \mathbf{PC}_T$, either $Sepset_{T,X} = NIL$ or $Sepset_{X,T} = NIL$ ***(Note that*** $\emptyset$ ***means empty set*** $\{\}$***, while*** NIL ***means Null pointer, i.e. there is no record for the corresponding subscript*** $(\boldsymbol{X}, \boldsymbol{Y})$**)**.

Proof. *Given each* $X \notin \mathbf{PC}_T$*, (1) If it is a non-descendant of* T*, it will be recognized as conditionally independent given some* $Sepset_{T,X} \neq NIL$*; (2) Else if it is a descendant of* T*, it may be falsely decided as conditionally dependent with* T*, which means that* $Sepset_{T,X} = NIL$*, and it will be contained in* $\mathbf{PC}_T^C$*; (3) Since we will call FindCanPC for each* $X \in \mathbf{PC}_T^C$*, if* $X \notin \mathbf{PC}_T$*,* T *will be recognized as conditionally independent given some* $Sepset_{X,T} \neq NIL$ *within FindCanPC(*X*). In short, for each* $X \notin \mathbf{PC}_T$*, it is always can be recognized conditionally independent given some set, and therefore* $Sepset_{T,X} \neq NIL$ *or* $Sepset_{X,T} \neq NIL$*.* ∎

Due that either $Sepset_{T,X} \neq NIL$ *or* $Sepset_{X,T} \neq NIL$, it is necessary to check them before the assignment as done at Line 15 of *IPC-MB*.

Lemma 3.8 Given the faithfulness assumption, $I(T, X|Sepset)$ is equal to say that all paths between T and X are blocked by $Sepset$, i.e. T is d-separated from X by $Sepset$.

Lemma 3.9 Given $X \in \mathbf{Sp}_T$ and $\mathbf{PC}_T^C$ = *FindCanPC*(T), $X \notin \mathbf{PC}_T^C$.

Proof. ***Theorem 3.2*** *tells that FindCanPC(*T*) won't output* T*'s non-descendants. Since* $\mathbf{Sp}_T \subseteq \mathbf{ND}_T$*, it means that* $\mathbf{Sp}_T \cap \mathbf{PC}_T^C = \emptyset$ *, i.e.* $X \notin \mathbf{PC}_T^C$ *given each* $X \in \mathbf{Sp}_T$*.* ∎

Theorem 3.4 Given $X \in \mathbf{PC}_T$ and $\mathbf{PC}_X^C$ =*FindCanPC*(X), for each $Y \in \mathbf{PC}_X^C$ but $Y \notin \mathbf{MB}_T$ and $Y \notin \mathbf{PC}_T^C$ (excluding processed and descendants of T if there are), if Y is conditionally dependent

with T given $Sepset_{T,X} \cup \{X\}$ or $Sepset_{X,T} \cup \{X\}$ (depending on which one is not NIL), Y is known as a true spouse of T.

Proof. *Given* $X \in \mathbf{PC}_T$, $Y \in \mathbf{PC}_X^C$ *but* $Y \notin \mathbf{MB}_T$ and $Y \notin \mathbf{PC}_T^C$, *it is secure to declare that T is connected with* X, *denoted as* $T - X$, *and T is NOT connected with Y ,denoted as* $Y \nsim T$. *Besides, due that* $Y \notin \mathbf{PC}_T$, *it is known that* $I(T, Y|Sepset)$ *where* $Sepset = (\text{Sepset}_{T,Y} \neq \emptyset)?\,\text{Sepset}_{T,Y} : \text{Sepset}_{Y,T}$ ***(Lemma 3.6** and **Lemma 3.7)** . In other words, Sepset blocks all possible paths connecting T and X (***Lemma 3.8***). To prove the statement, we have to study the following six cases separately considering that X may be a parent/child of T (*$X \in \mathbf{PC}_T$*) and Y can be a parent/child/descendant of X (*$Y \in \mathbf{PC}_X^C$*):*

1. $X \in \mathbf{Pa}_T$ and $Y \in \mathbf{Pa}_X$, i.e. $Y \in \mathbf{ND}_T$, $Y \to X \to T$ but $Y \nsim T$. To block the path $Y \to X \to T$, the statement that $X \in Sepset$ must be true. Otherwise, at least we have a non-blocked path, which is contradictory to the fact that $I(T, Y|Sepset)$and **Lemma 3.8**. Therefore, $Sepset \cup \{X\} = Sepse$,and $\sim I(T, Y|Sepset \cup \{X\})$ won't happen for this case;
2. $X \in \mathbf{Pa}_T$ and $Y \in \mathbf{Ch}_X$, i.e. $Y \leftarrow X \to T$ but $Y \nsim T$. Same proof as case 1;
3. $X \in \mathbf{Ch}_T$ and $Y \in \mathbf{Pa}_X$, i.e. $Y \to X \leftarrow T$ but $Y \nsim T$. It is easy to prove that adding X does will make the path $Y \to X \leftarrow T$ non-blocked, i.e. $Sepset \cup \{X\}$ won't d-separates Y and T anymore. Therefore, we have $\sim I(T, Y|Sepset \cup \{X\})$;
4. $X \in \mathbf{Ch}_T$ and $Y \in \mathbf{Ch}_X$, i.e. $Y \leftarrow X \leftarrow T$ but $Y \nsim T$. Same proof as case 1;
5. $X \in \mathbf{Pa}_T$ and $Y \in \mathbf{Des}_X$. (1) Since $Y \in \mathbf{PC}_X^C$ and $\mathbf{Des}_X$, there must exist, at least one, non-blocked path $Y - \cdots - X$. (2) Because $Y \notin \mathbf{PC}_T$, all paths connecting Y and T must be blocked by some $Sepset$. Assuming that there is one path $Y - \cdots - X$ known as open, then it is extendable to access T via X since $X \in \mathbf{Pa}_T$, e.g. $Y - \cdots - X \to T$. To ensure d-separation, this path $Y - \cdots - X \to T$ has to be blocked; therefore X has to be observed, i.e. $X \in Sepset$. Otherwise, $Y - \cdots - X \to T$ will keep open (since there is no chance to construct a converging pattern here with the existing of $X \to T$), which is contradictory to the fact that $I(T, Y|Sepset)$. Since $X \in Sepset$, it is impossible to have $\sim I(T, Y|Sepset \cup \{X\})$;
6. $X \in \mathbf{Ch}_T$ and $Y \in \mathbf{Des}_X$. Similar proof as case 5.

These six cases cover all possible happenings, so the proof itself is complete. From the discussion above, it is noticed that only the true spouse can satisfy $\sim I(T, Y|Sepset \cup \{X\})$ given $I(T, Y|Sepset)$, where $X \in \mathbf{PC}_T$, $Y \in \mathbf{PC}_X^C$ but $Y \notin \mathbf{MB}_T$ and $Y \notin \mathbf{PC}_T^C$. ■

Theorem 3.5 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution P faithful to a DAG G, all spouses of T of interest are found with IPC-MB.

Proof. $\mathbf{PC}_T$ (Line 12 of IPC-MB) *contains all the true parents and children, and* $\cup_{X \in \mathbf{PC}_T} \mathbf{PC}_X^C = \cup_{X \in \mathbf{PC}_T} \mathbf{Sp}_{T,X}^C = \mathbf{Sp}_T^C$ *contains all spouses of* T. *With each* $Y \in \mathbf{Sp}_T^C$ *but not in the current* $\mathbf{MB}_T$ *and not in* $\mathbf{PC}_T^C$, *it will be correctly recognized if it is true spouse (**Theorem 3.5**). Since this checking applies to all variables in* $\mathbf{Sp}_T^C$, we are able to find all spouses of T. ■

The determination of any true spouse is done in a manner different from the learning of parents/children. While searching for T's parents and children, we try to filter as many false positives as possible, reaching a set containing true parents and children, though some descendants are included as well. Then, those false positives are further filtered out, with only true positives left. However, while searching for the spouses of T, we directly check if each candidate is true or not.

Though the search of spouses proceeds in a different way, it depends on the output of *FindCanPC*, including spouse candidates and sepsets cached. This again reflects the importance of *FindCanPC*.

3.3 IPC-MB is Sound in Theory

Theorem 3.6 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution P faithful to a DAG G, the result induced by *IPC-MB* is $\mathbf{MB}_T$.

Proof. *IPC-MB is divided into two phases: learn the parents and children of T first, then further to learn T's spouses. The soundness of these two parts is demonstrated by* **Theorem 3.3** *and* **Theorem 3.5** *respectively. Both theorems show that not only all true members are ensured to be found and added into* $\mathbf{MB}_T$, *but no single false positive has chance to enter into* $\mathbf{MB}_T$. ■

So far, we have described in detail how IPC-MB induces the Markov blanket given a variable of interest, and we have proved that the output as returned by IPC-MB is correct. If we view the discussion so far as "qualitative" aspect about IPC-MB, in the next section 3.4, more "quantitative" features about IPC-MB will be studied.

3.4 Complexity Analysis

In IPC-MB, *FindCanPC* plays a critical role in the learning of $\mathbf{MB}_T$, and most computation happens inside *FindCanPC*. Therefore, we study the time and memory complexity of *FindCanPC* first, on which the overall cost can then be determined.

In algorithms like IPC-MB which depends primarily on a series of CI tests in the search, the overall measure of time complexity can be measured by the number of CI tests, i.e. $I_D(\cdot)$, as required.

Note: Our analysis is based on the assumption that there are enough data for thorough search as designated by the algorithm theoretically.

3.4.1 Time Complexity of FindCanPC

In this section, we focus on the performance of *FindCanPC* since it is viewed as the foundation of IPC-MB, and as we can see late, it determines the whole complexity of IPC-MB as well. Our discussion includes two aspects: qualitative and quantitative. The qualitative analysis gives us a "rough" picture of *FindCanPC*, leading us to explore more descriptive outcome.

Qualitative Analysis

Theorem 3.7 Given the assumption of faithfulness and correct conditional independence tests, any recognition of false positive in *FindCanPC* is achieved at the very first time.

Proof. *We can prove it by contradiction. Assume that (1) at the* ith*iteration,* $X \in \mathbf{PC}_T^C$, $\mathbf{S} \subset \mathbf{PC}_T^C$ *and* $I_D(T, X|\mathbf{S}) < \varepsilon$*, i.e.* $I(T, X|\mathbf{S})$ *and* X *is recognized as* $NonPC$*, one false positive; (2) there is* $\mathbf{S}' \subset \mathbf{S}$*, and* $I_D(T, X|\mathbf{S}') < \varepsilon$ *.With* $\mathbf{S} \subset \mathbf{PC}_T^C$ *and* $\mathbf{S}' \subset \mathbf{S}$ *, it is able to infer that* $\mathbf{S}' \subset \mathbf{PC}_T^C$*; then,* $\mathbf{S}'$ *must be met in earlier iteration, and* X *should has been removed from* $\mathbf{PC}_T^C$ *at iteration* $j(j < i)$ *when smaller conditioning set with the cardinality as* $\mathbf{S}'$ *is under study. This contradicts to the*

fact that $X \in \mathbf{PC}_T^C$ *at the beginning of the* i*th iteration. Therefore, if* X *is able to be recognized as false positive in FindCanPC, it must be found at the very first time.* ∎

Theorem 3.7 indicates that deleting any recognizable false positive in *FindCanPC* is achieved with the least cost**.** This theorem concludes the data-efficient feature of our work from the theoretical viewpoint, and we will revisit it again in next section to see if it applies globally within IPC-MB.

Quantitative Analysis

The most effective measure about the IPC-MB is the number of CI tests ($\mathrm{I_D}(\cdot)$ in our implementation)required considering that (1) IPC-MB and related algorithms depend on CI test to make decision, and (2) it is the most time consuming processing unit compared with other operations involved in the algorithm.

General Case

Our analysis starts with a very general scenario: Given T and $CanPC_T$, the search starts with empty conditioning set on, and it continues till there is no more CI tests left non-conducted, i.e. $|\mathbf{PC}_T^C| \leq cutSetSize$. The overall procedure is illustrated in 错误！未找到引用源。, and we assume totally there are K iterations.

Table 3.1 General analysis of the number of CI tests as required in *FindCanPC*.

Given T **and** $\mathbf{PC}_T^C$ **with** $\lvert\mathbf{PC}_T^C\rvert = M > 0$ (i.e at least one candidate neighbor)
Step 1: $cutSetSize = 0$, $\lvert\mathbf{PC}_T^C\rvert = M$ # of I_D Tests: $\binom{M-1}{0} * (M - N_0)$, *where* $N_0 = 0$ # of *Non-PC* Found: $N_1 (\geq 0)$
Step 2: $cutSetSize = 1$, $\lvert\mathbf{PC}_T^C\rvert = M - N_0 - N_1$ # of I_D Tests: $\binom{M-N_0-N_1-1}{1} * (M - N_0 - N_1)$ # of *Non-PC* Found: N_2
…
Step i: $cutSetSize = i - 1$, $\lvert\mathbf{PC}_T^C\rvert = M - N_0 - N_1 - \cdots - N_{i-1}$ # of I_D Tests: $\binom{M-N_0-N_1-\cdots-N_{i-1}-1}{1} * (M - N_0 - N_1 - \cdots - N_{i-1})$ # of *Non-PC* Found: N_i
…
Step K: $cutSetSize = K - 1$, $\lvert\mathbf{PC}_T^C\rvert = M - N_0 - \cdots - N_{i-1} - \cdots - N_{K-1}$ # of I_D Tests: $\binom{M-N_0 \cdots - N_{i-1} - \cdots - N_{K-1} - 1}{1} * (M - N_0 - \cdots - N_{i-1} \cdots -$

N_{K-1}) # of *Non-PC* Found: N_K
The learning terminates since $|\mathbf{PC}_T^C| \leq cutSetSize$ **after** $cutSetSize++$

Let $\sum_{CPC(X)}(CI)$ be the total number of CI tests as required in *FindCanPC(X)*, and the index $CPC(X)$ means the learning of candidate parents/children of X. To get it, we need summing the number of CI tests needed in each round:

$$\sum_{CPC(X)}(CI) = \binom{M-N_0-1}{0} * (M-N_0)+ \binom{M-N_0-N_1-1}{1} * (M-N_0-N_1)+\cdots+ \binom{M-N_0-\cdots-N_{i-1}-1}{i-1} * (M-N_0-\cdots-N_{i-1})+\cdots+ \binom{M-N_0-\cdots-N_{i-1}-\cdots-N_{K-1}-1}{K-1} * (M-N_0-\cdots-N_{i-1}-\cdots-N_{K-1}) \quad (3.1)$$

Where:

1. $N_i \geq 0 (i = 0..K)$
2. $i < M-N_0-\cdots-N_{i-1}$ $(1 \leq i < K-1)$ (the intermediate step)
3. $K \geq M-N_0-\cdots-N_{i-1}-\cdots-N_K$ (the terminating condition)

The ith $(i > 1)$ element can be simplified further:

$$\binom{M-N_0-\cdots-N_{i-1}-1}{i-1} * (M-N_0-\cdots-N_{i-1}) = \frac{(M-N_0-\cdots-N_{i-1}-1)!}{(M-N_0-\cdots-N_{i-1}-1-(i-1))!(i-1)!}(M-N_0-\cdots-N_{i-1}-1)$$

$$= \frac{(M-N_0-\cdots-N_{i-1}-1)!}{(M-N_0-\cdots-N_{i-1}-1-(i-1))!\,(i-1)!}(M-N_0-\cdots-N_{i-1}-1)$$

$$= \frac{(M-N_0-\cdots-N_{i-1})!}{(M-N_0-\cdots-N_{i-1}-i)!(i-1)!}$$

$$= \frac{(M-N_0-\cdots-N_{i-1})!*(i)}{(M-N_0-\cdots-N_{i-1}-i)!\,(i)!}$$

$$= i*\binom{M-N_0-\cdots-N_{i-1}}{i}$$

(3.2)

Then, replacing each item in equation (3.1) with (3.2), we get a compact representation:

$$\sum_{CPC(X)}(CI) = \sum_{i=1}^{K}\left(i * \binom{M - N_0 - \cdots - N_{i-1}}{i}\right)$$

(3.3)

Worst Case

From (3.3), we can infer that when each N_i equals to zero, $i * \binom{M - N_0 - \cdots - N_{i-1}}{i}$ is then maximized, so as the summing due that we may have maximum $K = M$ meanwhile. Then, we have new version of summing equation on the total number of CI tests as required:

$$\sum_{CPC(X)}(CI) = \sum_{i=1}^{M}\left(i * C_M^i\right) = M * 2^{M-1}$$

(3.4)

The example shown in Figure 3-5 is one such case satisfying (3.4). Since all attributes excluding T are T's children, none of them are conditionally independent from T given any conditioning set. Therefore, $N_i = 0$, and the loop has to continue from $cutSetSize = 0$ to $cutSetSize = M - 1$, totally M iterations.

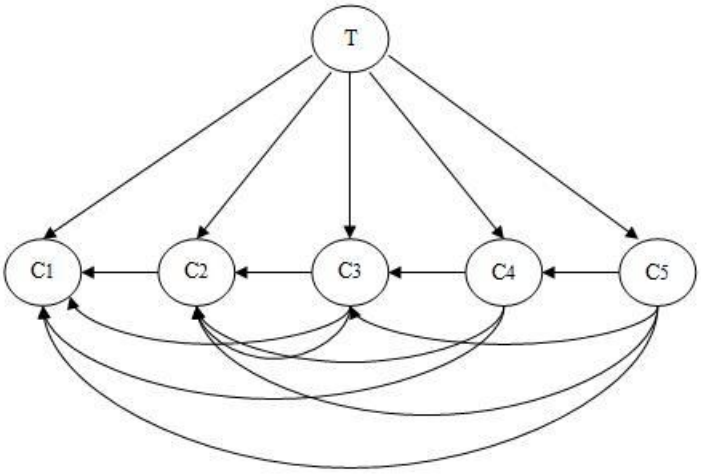


Figure 3-5: An example of network which has the largest size of Markov blanket, and *FindCanPC* performs the worst on it.

Simplest Case

Similarly, we infer that (3.3) is minimized when $M - N_1 \leq 1$, i.e. all recognizable false positives are found in the first iteration. Figure 3-6 also illustrates one such example in which T has one child, and all other attributes are spouses of T. Then, all spouses are conditionally independent with T given empty set, and all of them are deleted from $\mathbf{PC}_T^C$ by the end of the first iteration, with $\mathbf{PC}_T^C = \{C\}$ left. The loop, therefore, terminates since $|\mathbf{PC}_T^C| \leq cutSetSize = 1$. In this case, we have

$$\sum\nolimits_{CPC(X)} (CI) = \sum\nolimits_{i=1}^{M} \left(i * C_M^i\right) = 1 * C_M^0 = M \tag{3.5}$$

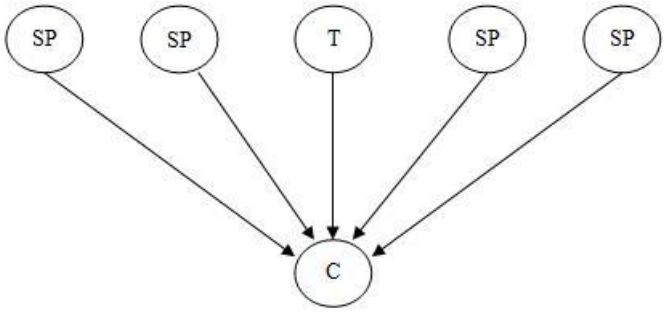


Figure 3-6: An example of network which has the largest size of Markov blanket, but FindCanPC perform the best on it.

Theorem 3.8 Given T and $CanPC_T$ of fixed cardinality (>1), the maximum number of CI tests as possibly required by *FindCanPC* is $M * 2^{M-1}$, and the possibly minimum amount is M , where $M = |\mathbf{PC}_T^C|$.

Theorem 3.8 gives the upper and lower bound of the time complexity of *FindCanPC*, and the actual amount is determined by the underlying topology. Fortunately, in applications, most false positives are removed given a small conditioning set, which results with actual cost far below $M * 2^{M-1}$ (refer to Section 4.5).

3.4.2 Time Complexity of IPC-MB

The measure of *FindCanPC* is the basis for the overall analysis of IPC-MB. Our analysis is composed of two steps: the learning of parents/children based on a sequential call of *FindCanPC*, as well as the induction of true spouses from spouse candidates as collected in the first step.

Analysis on the Learning of Parents/Children and Candidate Spouses

During the first phase of IPC-MB, *FindCanPC* is called repeatedly to achieve our goal, and the time complexity of this phase may be measured as how many times *FindCanPC* is called. Given **T** and **U**, a general equation can be constructed to reflect the time complexity:

$$\sum\nolimits_{CPC(T)}(CI)+\sum\nolimits_{X\in\mathbf{PC}_T^C}\left(\sum\nolimits_{CPC(X)}(CI)\right) \tag{3.6}$$

To maximize (3.6), we need not only maximize $\sum_{CPC(T)}(CI)$ and $\mathbf{PC}_T^C$ (the output of *FindCanPC(T)*), but $\sum_{CPC(X)}(CI)$ given each $X \in \mathbf{PC}_T^C$. The example of Figure 3-5 exactly satisfies these three conditions: (1) $\sum_{CPC(T)}(CI)$ is maximized as discussed in last section; (2) $\mathbf{PC}_T^C = \mathbf{U}\backslash\{T\}$, maximized meanwhile; and (3) each $\sum_{CPC(X)}(CI)$ is maximized due that all $\mathbf{U}\backslash\{X\}$ are parents or children of X. Therefore, by replacing each term of (3.6) with (3.4), we can infer that the maximum number of CI tests as required to induce the $PC(T)$:

$$\sum\nolimits_{\mathbf{PC}_T}(CI)=|\mathbf{U}|*\sum\nolimits_{i-1}^{|\mathbf{U}|-1}\left(i*\binom{|\mathbf{U}|-1}{i}\right)=|\mathbf{U}|^2*2^{|\mathbf{U}|-1} \tag{3.7}$$

where $\sum_{\mathbf{PC}_T}(CI)$ is newly introduced $|\mathbf{PC}_T^C|$ is replaced with $|\mathbf{U}|-1$.

Similarly, we know that (3.6) achieves the smallest value given the example of Figure 3-6:

$$\sum\nolimits_{\mathbf{PC}_T}(CI)=|\mathbf{U}|*(|\mathbf{U}|-1) \tag{3.8}$$

Note: Since the preparation of candidate spouses are done during this procedure without extra cost, no extra analysis is required on that.

Theorem 3.9 Given T and $\mathbf{U}$, the possibly maximum number of CI tests as required by IPC-MB to learn $\mathbf{PC}_T$ is $|\mathbf{U}|^2 * 2^{|\mathbf{U}|-1}$, and the minimum amount is $|\mathbf{U}| * (|\mathbf{U}| - 1)$.

In real applications, the actual cardinality of $\mathbf{MB}_T$ normally is much smaller than $\mathbf{U}$. Therefore, the actual cost generally is far below than the maximum value $|\mathbf{U}|^2 * 2^{|\mathbf{U}|-1}$. Besides, the underlying topology is influential to the actual performance, even though the underlying Markov blanket is known of size $|\mathbf{U}|$.

Analysis on the Learning of Spouses

Given $\mathbf{PC}_T$ and $\mathbf{Sp}_T^C$ ready, the determination of true spouses is done with a two-layer loop. Here we assume that the checking if $X \in \mathbf{X}$ can be done in constant time given hash-like storage, so the time complexity of the second phase of IPC-MB can also be measured with the number of CI tests (Line 15, IPC-MB).

To avoid repetition and save computing resource, we will ignore all $Y \in \mathbf{MB}_T$, i.e. those recognized and added to the $\mathbf{MB}_T$ container already. Besides, considering that (1) those positive ones in $\mathbf{PC}_T^C$ are added to $\mathbf{MB}_T$, and (2) those false positives as contained in $\mathbf{PC}_T^C$ can only be T's descendants, the whole $\mathbf{PC}_T^C$ will be ignored. Therefore, we only consider $Y \in \mathbf{Sp}_T^C \backslash \mathbf{MB}_T \backslash \mathbf{PC}_T^C$. The total number of CI tests as required for the recognition of true spouses can then be denoted as:

$$\sum\nolimits_{\mathbf{Sp}_T} (CI) = \sum\nolimits_{X \in \mathbf{PC}_T} |\mathbf{Sp}_{T,X}^C \backslash \mathbf{MB}_T \backslash \mathbf{PC}_T^C| \tag{3.9}$$

Assuming that $|\mathbf{PC}_T|/|\mathbf{U}|$=a, where $0 \leq \mathrm{a} \leq 1$. If $\mathrm{a} = 1$, obviously nothing left to do; otherwise, we have $|\mathbf{Sp}_T^C \backslash \mathbf{MB}_T \backslash \mathbf{PC}_T^C|/|\mathbf{U}| < (1 - a)$ considering that $\mathbf{MB}_T$ may increases with time on and $\mathbf{PC}_T \subseteq \mathbf{PC}_T^C$. Therefore, at most $(1 - \mathrm{a}) * |\mathbf{U}|$ times of CI tests are available in each inner loop, and maximally, we need $\mathrm{a}(1 - \mathrm{a}) * |\mathbf{U}|^2$ CI tests by summing $\mathrm{a} * |\mathbf{U}|$ times of $(1 - \mathrm{a}) * |\mathbf{U}|$ based on (3.9).

Theorem 3.10 Given T and $\mathbf{U}$, maximally $\frac{1}{4}|\mathbf{U}|^2$ CI tests are required in the second phase of IPC-MB to learn true spouses.

Proof. $\mathrm{a}(1 - \mathrm{a}) * |\mathbf{U}|^2$ *achieves the maximum value when* $\mathrm{a} = 0.5$*, i.e.* $\frac{1}{4}|\mathbf{U}|^2$. ∎

So, when $|\mathbf{PC}_T|/|\mathbf{U}|$=0.5, i.e. half of $\mathbf{U}$ are parents and children of T , the second phase of IPC-MB needs the maximum number of CI tests. However, compared with the worse case of the first phase, $|\mathbf{U}|^2 * 2^{|\mathbf{U}|-1}$, $\frac{1}{4}|\mathbf{U}|^2$ is relatively small, and its increasing rate is much slower than the previous one. Therefore, the worst complexity of IPC-MB is determined more by the first phase. Interestingly, given the examples of Figure 3-5, no additional CI tests are required in the second phase since $\mathbf{MB}_T = \mathbf{PC}_T = \mathbf{U}$.

Theorem 3.11 The worst performance of IPC-MB on time efficiency is $|\mathbf{U}|^2 * 2^{|\mathbf{U}|-1}$, and the best performance is less than $\frac{5}{4}|\mathbf{U}|^2 - |\mathbf{U}|$.

Proof. *The worst case is discussed. The best performance is determined by the topology, so we only give a loose upper bound by summing the best case of the first phase* $|\mathbf{U}| * (|\mathbf{U}| - 1)$ *and the worst case of the second phase* $\frac{1}{4}|\mathbf{U}|^2$. ■

3.4.3 Memory Requirement of FindCanPC

The access to memory (RAM) is known as much faster than disk I/O operation, so ideally we want all data to be referred during the computing available in memory. Though this is impossible for most cases, we prefer some solution to reduce the frequency of disk access as much as possible. In this and next sections, we analysis the memory consumption of *FindCanPC* and *IPC-MB* respectively, based on our own implementation strategy.

NonPC

It is used to cache those recognized as false positive in each iteration, and it is trivial to know that its maximum size won't exceed $|\mathbf{PC}_T^C|$.

Subsets S and Contingency Tables

The number of subsets and the size of each subset are known upon entering the innermost loop (Line 5, Figure 3-1), so we prefer to allocate memory to cache all $\mathbf{S} \subseteq \mathbf{PC}_T^C \backslash \{X\}$ of the size $cutSetSize$ between the Line 5 and Line 11. Given $\mathbf{PC}_T^C$ and $cutSetSize$, the number of subsets of is $\mathbf{PC}_T^C \backslash \{X\}$ is

$$\binom{cutSetSize}{|\mathbf{PC}_T^C| - 1}, \text{ where } 0 \leq cutSetSize \leq |\mathbf{PC}_T^C| - 1$$

Considering that the cardinality of each subset is of size $cutSetSize$, and assuming that each variable is binary (including the target), each contingency table corresponding to $I_D(T, X|\mathbf{S})$ has $2^{2+cutSetSize}$ cells. Then totally we need

$$2^{2+cutSetSize} * \binom{cutSetSize}{|\mathbf{PC}_T^C| - 1} \tag{3.10}$$

In (3.10) it is noticed that there are three factors to influence the actual complexity, including $|\mathbf{PC}_T^C|$, $cutSetSize$ and the number of values of each variable. In fact, the first two factors are influenced by the actual topology, and the third one is determined by the actual problem. In our implementation, hash table-like container is used to hold contingency table considering there may exist empty cells, i.e. cells with no values. However, we have to admit that our current implementation cannot deal with too large scale of problems on a common PC machine.

$Sepset_{T,X}$

This is global allocation since it needs to be referred later in IPC-MB.

Lemma 3.10 Those seprator set (i.e. $Sepset$ as found in our algorithm) as found and cached for T and X in IPC-MB, i.e. $Sepset_{T,X}$ or $Sepset_{X,T}$, is the minimal such set.

Proof. *It can be proved by contradiction. Assuming that we find* $Sepset^*_{T,X}$ *such that* $Sepset_{T,X} \subset Sepset^*_{T,X}$, *and* $I(T, X|Sepset^*_{T,X})$. *Given* $|Sepset_{T,X}| < |Sepset^*_{T,X}|$, *it means that* $Sepset_{T,X}$ *is missed though it appears at an early time in FindCanPC, which happens only when* $I_D(T, X|Sepset_{T,X}) > \varepsilon$. *Obviously, it is contradictory with the fact that* $Sepset_{T,X}$ *is a valid separator set.* ∎

Given each pair of (T, X), since we will cache only one separator set to satisfy $I_D(T, X|Sepset) \leq \varepsilon$, the total number of such seprator won't exceed $|\mathbf{PC}_T^C|$. However, the size for $Sepset_{T,X}$ and $Sepset_{T,Y}$ may differ since they may be recognized in different iteration. Then, we can denote the total memory as required by caching $Sepset$ as:

$$\sum_{I_D(T,X)\leq\varepsilon} |Sepset_{T,X}|, \text{ with restrict to } 0 \leq |Sepset_{T,X}| \leq |\mathbf{PC}_T^C| - 1$$

Empty $Sepset_{T,X}$ appears when X is found conditionally independent of T given empty set in the first iteration; if no variable in $\mathbf{PC}_T^C$ is removed until the iteration $|\mathbf{PC}_T^C| - 1$, we may have

$Sepset_{T,X}$ as large as $|\mathbf{PC}_T^C| - 1$. Therefore, the actual memory footprint is influenced by the topology as well.

Theorem 3.12 The memory allocation for $Sepset_{X,Y}$ is minimized in IPC-MB.

Proof. ***Lemma 3.10*** *tells us that* $Sepset_{X,Y}$ *as found is the minimal one; besides, it is known that we only cache either* $Sepset_{T,X}$ *or* $Sepset_{X,T}$*, then it is trivial to know that the memory allocation for this part is minimized in IPC-MB.* ∎

3.4.4 Memory Requirement of IPC-MB

Based on our discussion of *FindCanPC*, it is known that only $Sepset_{X,Y}$ is global allocation. In addition to this, all other allocation in *FindCanPC* becomes free upon leaving it.

Considering that there is no other large memory requirement in IPC-MB, or they are relatively small as compared with *FindCanPC*, no more space is left for this discussion.

3.4.5 Brief Conclusion on the Complexity of IPC-MB

By making full use of the underlying topology information, IPC-MB learns the Markov blanket of T via iterative local search. Within each local search round, it takes the strategy of removing any false positive at the first moment it is found, which is very different from all previous work and is expected to be much more efficient than them. Its "smart" strategy makes the overall architecture very simple, easy to understand and implement as well.

Our analysis is built around this design as well. As its name indicates, the overall cost, no matter time or memory, is determined by that of *FindCanPC*. However, the actual complexity will be influenced by the underlying actual topology. From our analysis, it is observed that the theoretically best and worst cases have very different performance, ranging from linear to exponential growth.

3.5 Data Efficiency and Reliability of IPC-MB

Data efficiency is critical for the practical value of one algorithm since instances available for training or learning are limited very often. In algorithms built on statistical testing, like the CI test employed in IPC-MB, normally the fewer variables involved in $I_D(X, Y|\mathbf{Z})$, the more data

efficient is the algorithm. This is because CI test error, being the primary source of error, is the result of unnecessary large condition set leading to the curse-of-dimensionality or choosing an inaccurate conditioning set due to partial information. Therefore, data efficiency indirectly influences the accuracy. In fact, in Chapter 4, we can observe obvious difference in the actual accuracy between algorithms with low and high data efficiency.

In this section, we discuss the data efficiency of IPC-MB from theoretical viewpoint, and we will revisit this topic in Chapter 4 and Chapter 5 considering that and it is so important and will never be overemphasized. In fact, you will find in our conclusion later (Section 5.4) that data efficiency is the most merit of IPC-MB, which permits IPC-MB not only to be very time efficient but to achieve the highest accuracy as compared with similar works.

Lemma 3.11 Any false positive as recognized in *FindCanPC* is conditioned with the smallest conditioning set.

Proof. *Please refer to the proof of* **Lemma 3.10**. ∎

Lemma 3.12 Any false positive as contained in $\mathbf{PC}_T^C$ is recognized with the smallest conditioning set.

Proof. *The recognition of any false positive X is via the call of FindCanPC(X) which tells us if* $T \in \mathbf{PC}_X^C$. *Those false positives relative to X are recognized with the smallest conditioning set (***Lemma 3.11***), so we can say that any false positive of* $\mathbf{PC}_T^C$ *is recognized with the smallest conditioning set.* ∎

Theorem 3.13 Given T and $\mathbf{U}$, the recognition of any $X \in \mathbf{PC}_T$ is achieved with the least conditioning set.

Proof. *For any* $X \in \mathbf{PC}_T^C$ *as output by FindCanPC(T), it enters into or leaves from (by deleted)* $\mathbf{PC}_T^C$ *according to CI tests with the smallest conditioning set, as ensured by* ***Lemma 3.11*** *and* ***Lemma 3.12****. Therefore, we can declare that the recognition of any* $X \in \mathbf{PC}_T$ *is achieved with the least conditioning set.* ∎

Lemma 3.13 The recognition of true spouses is achieved with the least conditioning set.

Proof. *The recognition of any true spouse is realized with two phases: (1) the preparation of candidate spouses, and (2) the determination of one true spouse. Regarding the preparation of*

*candidate spouses, it is learned by a series of FindCanPC(X), so it is known that the smallest conditioning set is used (**Lemma 3.11**). Besides, the Sepset as used to determine if a candidate is true spouse is also minimized in cardinality (**Lemma 3.10**), therefore, the recognition of any true spouse is achieved with the least conditioning set.* ∎

Theorem 3.14 No algorithm can be more data efficient than IPC-MB.

Proof. ***Theorem 3.13*** *and* ***Lemma 3.13*** *ensure the least conditioning set is used to recognize parents/children and spouses respectively, so it is known that the recognition of* $\mathbf{MB}_T$ *realizes the best performance in term of data efficiency.* ∎

In real application, we may only have limited data for learning. To ensure the trustability of statistical testing like $I_D(X, Y|\mathbf{Z})$, we expect to make the conditioning set $\mathbf{Z}$ as small as possible. Given the same amount of training data, a test with a smaller conditioning set will always produce a more trustable outcome than the one with larger one. Considering that IPC-MB will resort to the smallest conditioning set with priority, it is believed that the output of IPC-MB is more reliable compared to similar algorithms given the same scale of data for training. Note that this merit of IPC-MB is not exchanged with any sacrifice of correctness, which again makes IPC-MB attractive.

3.6 Analysis of Special Case: Polyrtree

Being a special case of Bayesian network, a polytree is a directed acyclic graph with the property that ignoring the directions on edges yields a graph with no undirected cycles[2]. In other words, there exists unique path between each possible couple of nodes (see Figure 3-7 for an example), so polytree is the "thinnest" Bayesian network. In this section, we will discuss the expected behavior of IPC-MB given a polytree, and experiments on a polytree-like Bayesian network are included in Chapitre 4.

Theorem 3.15 Given a polytree network, the call of *FindCanPC(X)* in IPC-MB will output the exact $\mathbf{PC}_X$, under the faithfulness and correct CI test assumptions.

Proof. *From* ***Theorem 3.1*** *and* ***Theorem 3.2****, it is know that FindCanPC(X) will output a superset of* $\mathbf{PC}_X$*, and the only possible false positives are X's descendants. Therefore, here we only need to prove that descendants of X won't be falsely output by FindCanPC given a polytree*

network, and this can be proved by contradiction. Assume some $Y \in \mathbf{Des}_X$ *appears in the output of FindCanPC(X). Because there is only one directed path from X to Y in the polytree, it must be either* $X \to Z \to Y$ *or* $X \to Z \to \cdots \to Y$ *, where Z is X's child. It is trivial to know that Z blocks this unique path from X to Y due to the head-to-tail connection, plus the fact that Z will always stay in* $CanPC_T$ *and output by FindCanPC, hence Y won't pass the test* $I_D(X, Y|Z)$ *and will be removed successfully. This proof applies to all X's descendants, so none of them will appear in the output of FindCanPC(X).* ∎

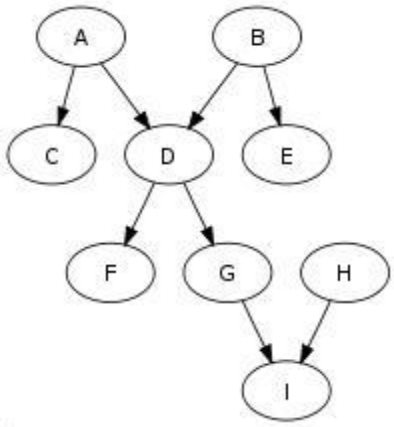


Figure 3-7: A simple example of polytree. The original graph can be found online at http://en.wikipedia.org/wiki/Polytree.

Theorem 3.16 Under the faithfulness and correct CI test assumptions, given a polytree network, all $Y \notin \mathbf{PC}_X$ will be recognized by some conditional independence tests $I_D(X, Y|\mathbf{S})$ where $|\mathbf{S}| \leq 1$, in *FindCanPC*(*X*).

Proof. *Given any* $Y \notin \mathbf{PC}_X$*, there exists a unique path from Y to X in a polytree network, and it can be one of the following three cases:* $Y \to \cdots \to Z \to X$ *,* $X \to Z \to \cdots \to Y$ *, and* $Y \to \cdots \to Z \leftarrow X$*. In the first two cases (linear connection), Y is known as independent of X given Z , or we say Z blocks the path. And in the third case (converging connection), Y is known as independent of X given empty set. Hence, all non-parents/children of X will fail some* $I_D(X, Y|\mathbf{S})$ where $|\mathbf{S}| \leq 1$. ∎

Corollary 3.1 In *FindCanPC(X)*, all false positives will be removed from $\mathbf{PC}_T^C$ in the loop of $cutSetSize = 0$ or $cutSetSize = 1$.

Proof. *It is trivial to infer this from* ***Theorem 3.16.*** ∎

Corollary 3.2 In *FindCanPC(X)*, spouses are removed from $\mathbf{PC}_T^C$ in the loop of *cutSetSize* = 0.

Proof. *It is trivial to infer this from* ***Theorem 3.16.*** ∎

With **Corollary 3.1** and **Corollary 3.2**, we can infer three points as below:

1. Computing complexity is greatly reduced since all false positives are recognized and removed in the first two iterations (Line 3-16, *FindCanPC*, Figure 3-1);
2. The decision to remove these false positives is made given small conditioning set, so the decision as made according to the CI testing result is trustable, which is valuable in practice;
3. Since each spouse Y is recognized with empty separator, i.e. $Sepset_{X,Y} = \emptyset$, the conditional test involved to recogze a true spouse in IPC-MB (Line 15, Figure 3-3) will have conditioning set of size one only. Like the second point, it reflects the data efficiency of IPC-MB as well.

Even though we can remove all false positives in the first two iterations within *FindCanPC(X)*, it doesn't mean that only two iterations are needed. We have to continue the search until $|\mathbf{PC}_T^C| \leq cutSetSize$, hence, the actual time and memory complexity are influenced by the size of $\mathbf{PC}_X$.

3.7 Parallel version of IPC-MB

3.7.1 Overall illustration

Though IPC-MB is proposed to be more time efficient than PCMB, it still could be very computationally intensive in the worst case. Considering performance is critical, we devote a small section to discuss the parallel version of IPC-MB.

Given the output of *FindCanPC*(T), i.e. $\mathbf{PC}_T^C$, the remaining processing (Line 4-19) in IPC-MB, actually, can proceed in parallel since each branch is independent one another, as demonstrated in Figure 3-8, where $\mathbf{PC}_T^C = \{X_1, \dots, X_k\}$.

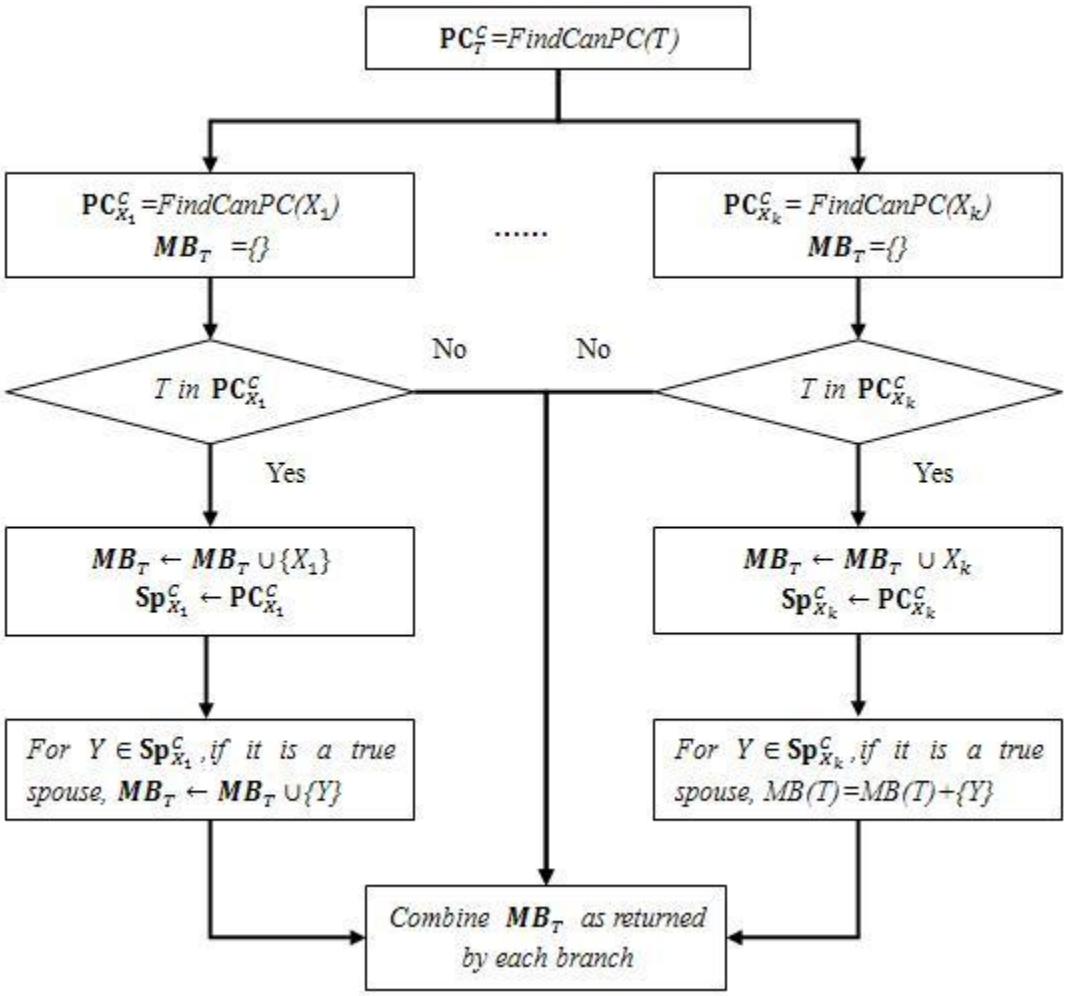


Figure 3-8: Parallel version of IPC-MB.

3.7.2 Proof of soundness

Theorem 3.17 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution P faithful to a DAG G, Parallel version of IPC-MB, denoted as Parallel-IPC-MB produces the same output of IPC-MB, if given the same inputs.

Proof. *(1)It's trivial to know that FindCanPC(T) produces the same result as it works in IPC-MB.(2)Then, for each* $X_i \in \mathbf{PC}_T^C$, *FindCanPC(*X_i*) is called separately, and it produces the same result as in sequential version since each FindCanPC(*X_i*) is fed with the same inputs in both versions. (3)That* $X_i \in \mathbf{PC}_T$ *if* $T \in \mathbf{PC}_{X_i}^C$ *still be true considering that both* $\mathbf{PC}_T^C$ *and* $\mathbf{PC}_{X_i}^C$ *are what expected as in the non-parallel version. (4) Then* $\mathbf{Sp}_{T,X}^C$ *is also the same as that in IPC-MB.*

(5) The remaining checking of $Y \in \mathbf{Sp}^{C}_{T,X}$ *is expected to produce the same result if both* $Sepset_{T,X}$ *and* $Sepset_{X,T}$ *are ready in each branch then. Obviously, this is true because FindCanPC(T) and FindCanPC(*X_i*), in which both* $Sepsets$ *are collected respectively, have been conducted by then.*

Therefore, both ***Theorem 3.3*** *and* ***Theorem 3.5*** *still work for Parallel-IPC-MB, and it is expected to produce the same outcome as IPC-MB given the same assumptions and inputs.* ∎

3.7.3 Time and space complexity

Theorem 3.18 The worst performance of Parallel-IPC-MB on time efficiency is

$$\mathbf{2*|U|} * 2^{\mathbf{|U|}-1} = \mathbf{|U|} * 2^{\mathbf{|U|}}$$

Proof. *Since the timing consumption of FindCanPC(T) is not avoidable, and the remaining FindCanPC(X) proceeds in paralle, totally we need only consider two times of the worst case of FindCanPC. Then, this result is trivial to infer on the basis of* ***Theorem 3.11****.* ∎

Regarding the memory consumption, there is no gain in the Parallel-IPC-MB since *FindCanPC* runs in a serial order in IPC-MB, and it has the same effect as the memory allocation on each machine in the parallel version, assuming each branch is distributed to one individual machine for processing.

3.7.4 About implementation

The training data has to be copied to each machine where we want the individual branch to run. Though it may be time consuming as transferred via network, the cost may be worthy in large scale of processing where the transferring time may be ignorable as compared with what we save on the learning.

In fact, it is believed that there must exist more fine-grained parallel versions of IPC-MB, but it is not the focus of this project and won't be discussed further here.

3.8 Conclusion

Given the faithfulness assumption, $\mathbf{MB}_T$ is known to contain T's parents, children and spouses. Among them, parents and children are directly connected to T, and spouses connect as well as

point to T's child(ren). With these topology information and **Theorem 1.4** in mind, constraint-based learning is believed to be more suitable than score-and-search based approach because it doesn't have to explore among the complete space, but in a greatly reduced sub-space. IPC-MB achieves this goal quite well by filtering out as many $X \notin \mathbf{PC}_T$, also as early, as possible, and with economical cost. Besides, by determining if $I(X, Y|\mathbf{Z})$ with as small conditioning set $\mathbf{Z}$ as possible, more reliable performance is expected given limited data in practice.

IPC-MB is the core part of this thesis project, and more applications derived from it will be discussed in the following text. Before that, empirical study will be presented in Chapter 4 to confirm what we have presented, and a comprehensive comparison is included in Chapter 5 between IPC-MB and other representative works.

Chapitre 4 EMPIRICAL STUDY OF MARKOV BLANKET LEARNING

4.1 Experiment Design

In Chapitre 3, we explain how IPC-MB works, prove its correctness, and analyze its expected performance on time and space complexity, and data efficiency. In this chapter, a series of experiments are conducted to compare the relative performance of IPC-MB, IAMB, PCMB and PC algorithms.

We only compare our algorithm with two typical Markov blanket learning algorithms, IAMB and PCMB, considering that (1) both of them are proved correct, and highly referred; (2) IAMB is the best known and also simple and time efficient; (3) PCMB is the latest published work, and it represents a new direction of this research field. Though MMPC/MB and HITON-PC/MB appear before PCMB, they are proved not correct and hence ignored. In additional to IAMB and PCMB, we also include PC algorithm to allow us to observe the difference between global and local learning algorithms.

In the experiments, five networks are used. Three of them are known benchmark examples, including Asia [39] with 8 nodes, Alarm [40] with 37 nodes and Hailfinder network with 56 nodes [41]. The other two are artificially created networks shipped with BNJ package[42], one has 152 nodes and the other is a polytree derived from Alarm. For easy reference, they are named Test152 and PolyAlarm respectively. Data are sampled from these five networks, and be fed to algorithms to recover the underlying network. We run IAMB, PCMB and IPC-MB with each node in the BN taken in turn as the target variable T and report the average performance over multiple rounds. Our discussion contains accuracy, time efficiency and data efficiency.

In Section 4.2, we introduce briefly data sets used. From Section 4.4 to Section 4.6, experimental results as well as some conclusions as derived are presented. We conclude briefly in Section 4.7.

4.2 Data Sets

The five selected networks represent four types of typical problem we are interested to study:

- Small problem, e.g. Asia;
- Medium size application, e.g. Alarm;

- Larger scale problem, e.g. Hailfinder and Test152. Though Hailfinder has only 56 nodes, its nodes have up to 11 values, so its search space actually is quite large. As we can see later, all four algorithms have difficulties to produce satisfactory results with 20,000 instances; while, acceptable results are observed on Test152 with only 2,500 instances;
- Polytree is a topology with at most one undirected path between any two vertices which allows more efficient computations and is a good example to study the data efficiency of our algorithm.

Asia

Asia is a small Bayesian network linking tuberculosis, lung cancer or bronchitis respectively and different factors, for example whether or not the patient has been to Asia recently. It firstly appeared in Lauritzen and Spiegelhalter's work [39], 1988, and has been widely referred in the past two decades. Its structure and the corresponding CPTs are illustrated in Figure 4-1.

Although small, this network allows us 1) to have a look at the corresponding performance of the four algorithms given a simple problem, and 2) show that the four algorithms may all output the correct outcomes given enough data.

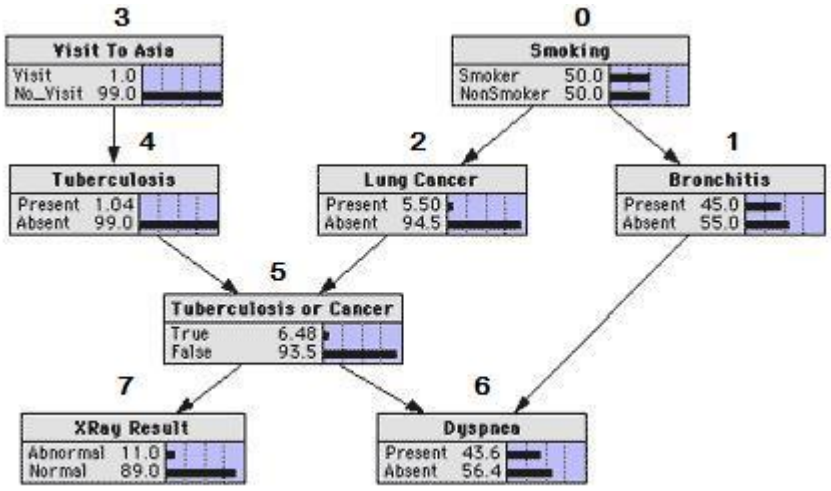


Figure 4-1: Asia Bayesian Network including 8 nodes of two states and 8 arcs, along with its CPTs. For reference purpose, each node is assigned one unique ID, from 0 to 7. The original graph can be found at http://www.norsys.com/netlib/asia.htm.

ALARM

ALARM stands for ‘A Logical Alarm Reduction Mechanism,’ a network for monitoring patients in intensive care. It was first introduced by Beinlich et al. in 1989 [40], and it consists of 37 nodes of two, three or four states, and 46 arcs. It is a commonly viewed as a representative of a real life Bayesian network.

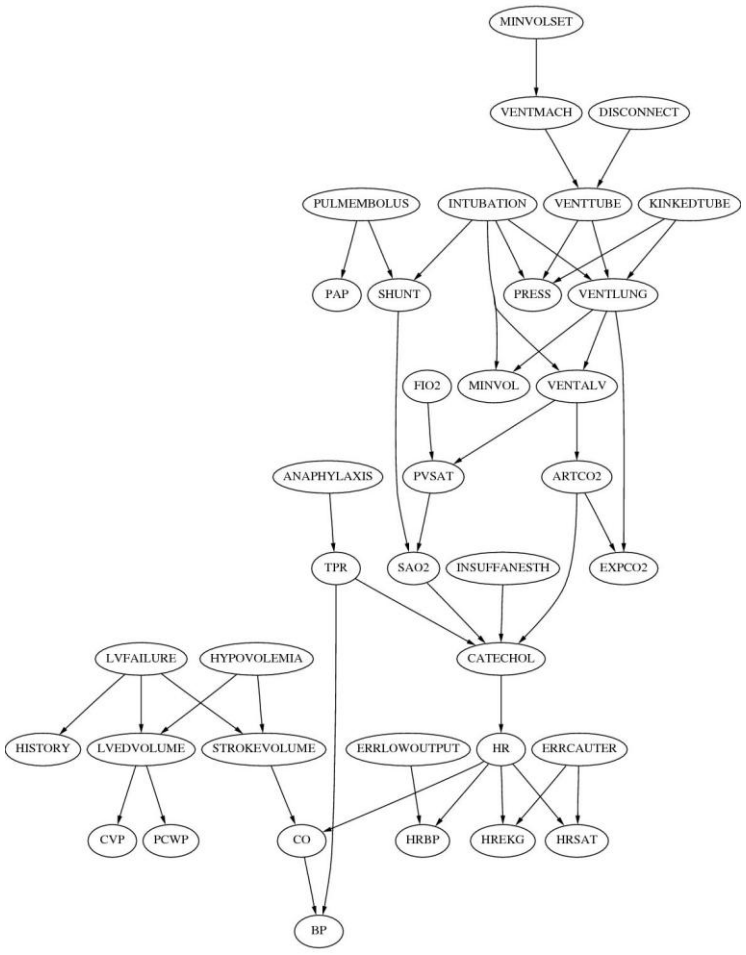


Figure 4-2: Alarm Bayesian Network including 37 nodes of two, three or four states (To save space, the CPTs are ignored). The original graph can be found online at http://compbio.cs.huji.ac.il/Repository/Datasets/alarm/alarm.htm.

ALARM of Polytree Version

This polytree version Alarm network, including the structure and parameters, is included in the installation package of Bayesian Network tool in Java (BNJ) [42]. This network is denoted as PolyAlarm for short in the remaining text.

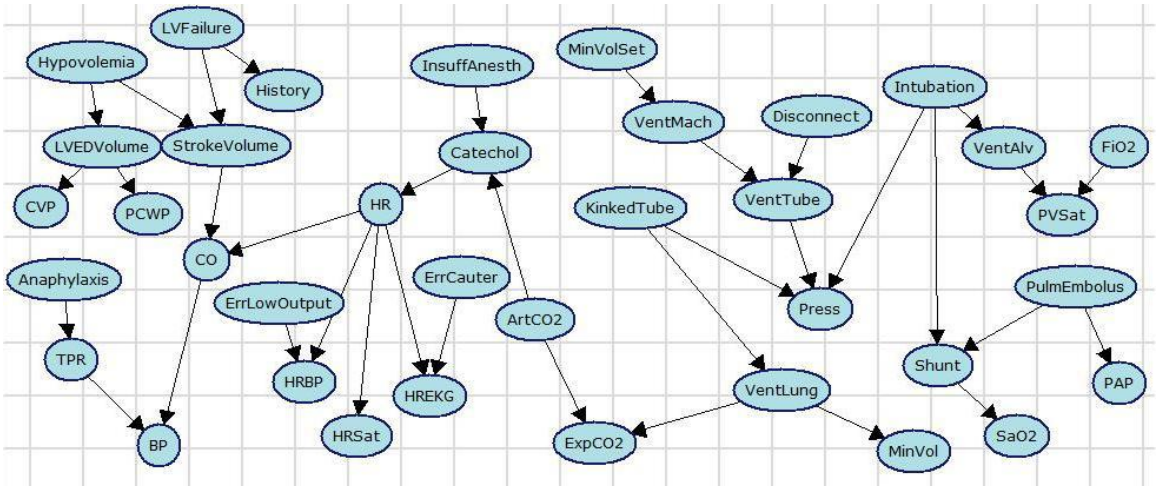


Figure 4-3: A polytree derived from Alarm Bayesian Network [40]. This graph is created by BNJ tool.

Hailfinder

Hailfinder [41] is a Bayesian system that forecasts severe summer hail in Northeastern Colorado. It is the first such system to apply Bayesian models in the realm of meteorology. Hailfinder contains 56 nodes, and the nodes contain two to eleven different values. Compared with Alarm in which four-value is the maximum, the underlying search space of Hailfinder is much larger.

Test152

This network is shipped with the installation package of BNJ tool, as a testing example. Since it contains 152 binary nodes, it is called as Test152 by us for quick reference.

Summary

The following table and figure summarize the features of the five networks under experiment. In Table 4.1, the total number of nodes and arcs, the largest size of Markov blanket as contained, and the number of states of nodes about Asia, Alarm, PolyAlarm, Hailfinder and Test152 are presented. The distribution about the cardinalities of Markov blankets as contained in the five networks are illustrated in Figure 4-4, and it is observed that even the largest Markov blankets are much smaller than the whole Bayesian networks in size. Therefore, we can conclude that (1) feature selection is necessary to remove non-related attributes; (2) real networks are mostly sparse, as illustrated in Figure 3-5.

Table 4.1: Feature summary of data sets

Bayesian Network	Values of Nodes	# of Nodes	# of Arcs	Size of largest MB
Asia	2	8	8	5
Alarm	2/3/4	37	46	8
PolyAlarm	2/3/4	37	36	8
Hailfinder	2/3/4/5/6/7/11	56	66	17
Test152	2	152	200	5

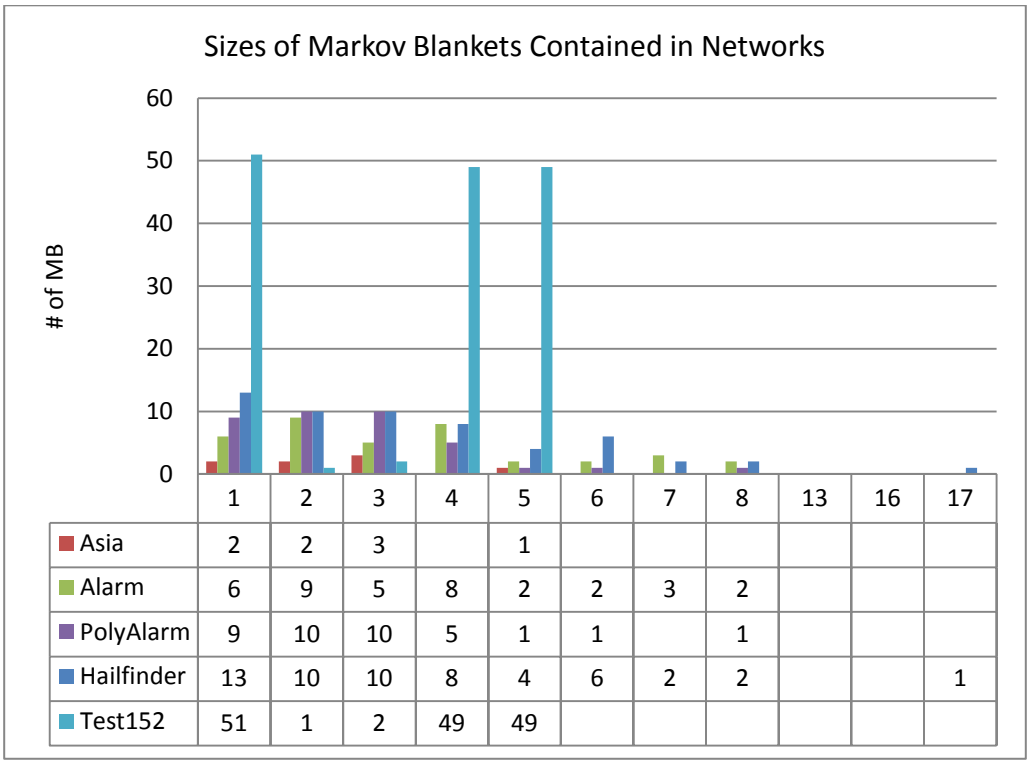


Figure 4-4: Distribution of the size of Markov blankets as contained in Asia, Alarm, PolyAlarm, Hailfinder and Test152.

4.3 Implementation Version of IPC-MB

In Section 2.2, we have stated that we ignore CI test for which there are not enough instances to ensure trustable result. In practice, the common choice of k in the inequality (1.5) is 5, and it is followed in our implementation as well. Besides, in *FindCanPC*, if there is no reliable CI test available in the loop of $cutSetSize$, no further search will be conducted, and the learning

terminates because all further tests will be of higher degree due to the growing conditioning set size. Figure 4-5 illustrates the implemented version of *FindCanPC*.

Similar strategy is taken in the implementation of IAMB, PCMB and PC algorithms, allowing for a fair comparison.

```
FindCanPC(T: Target, PC_T^C: Candidate PC, D: Dataset, ε: SignificanceValue)
{
1.  NonPC = ∅;
2.  cutSetSize = 0;
3.  repeat
4.    notReliableAnyMore = true;
5.    for( ∀X ∈ PC_T^C) do
6.      for(S ⊆ PC_T^C \{X}) with |S| = cutSetSize) do
7.        if(I_D(T, X|S) is reliable)then
8.          notReliableAnyMore = false;
9.          if(I_D(T, X|S) ≤ ε) then
10.           NonPC = NonPC ∪ {X};
11.           Sepset_{T,X} = S;        //Cache for later reference
12.           break;
13.         end if
14.       end if
15.     end for
16.   end for
17.   PC_T^C = PC_T^C \ NonPC;
18.   NonPC = ∅;
19.   cutSetSize ++;
20. until (|PC_T^C| ≤ cutSetSize  or notReliableAnyMore = true)
21. return PC_T^C;
}
```

Figure 4-5: The implemented version of FindCanPC that considers reliability of statistical tests. Its original version can be found in Figure 3-1, and the differences are illustrated in bold here for comparison convenience.

4.4 Accuracy

The experiments in this section focus on the accuracy of the algorithms. We run IAMB, PCMB and IPC-MB with each node in each BN as the target variable T and then, report the average

precision and recall over all the nodes for each BN. **Precision** is the number of true positives in the output divided by the number of nodes in the output. **Recall** is the number of true positives in the output divided by the number of true positives in the BN. We also combine precision and recall as

$$distance = \sqrt{(1 - precision)^2 + (1 - recall)^2} \tag{4.1}$$

to measure the Euclidean **distance** from perfect precision and recall. The significance level (ε) for the independence test is 0.05. These experimental specifications follow that of [13], with the expectation of comparable results. PC algorithm is ran a single time given each data set to induce the whole network, and the precision, recall and distance are measured similarly over each node. *Note: for each sample size, we prepare 10 to 20 groups of data for multiple-folder simulation.*

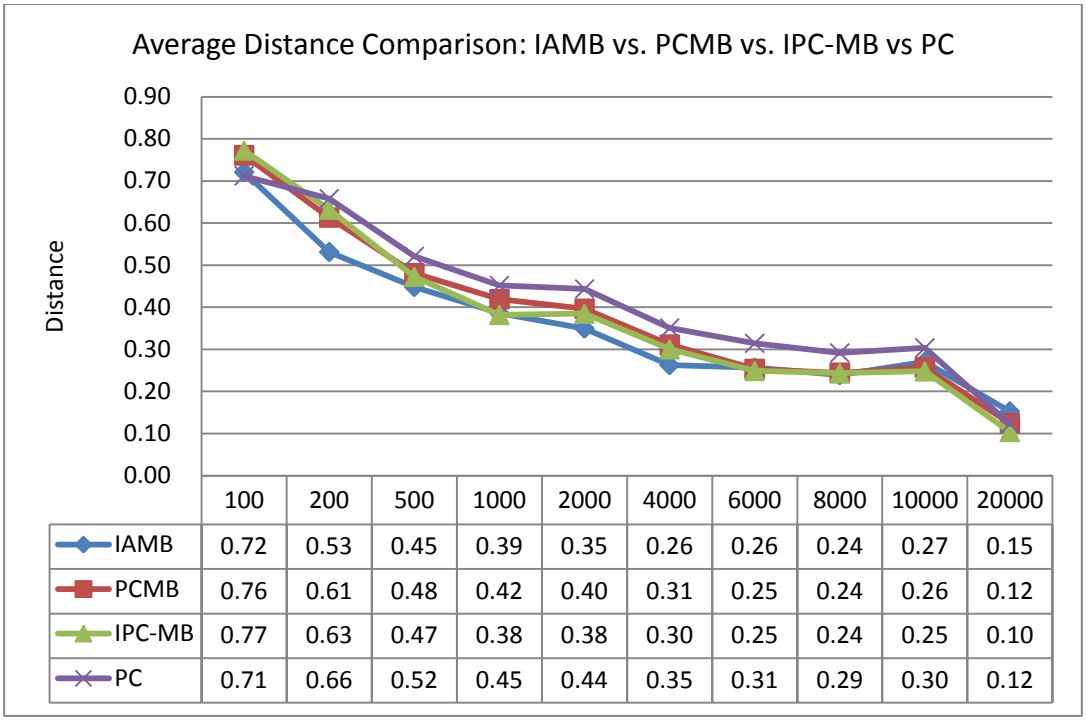
4.4.1 Small Network: Asia

Asia is a very small network with only eight nodes, and two of them have Markov blanket of size one. Because Asia is small, we used a 20-folds simulation experiment and report the average and standard deviations (Table 4.2).

Table 4.2: Accuracy comparison of IAMB, PCMB, IPC-MB and PC over Asia network.

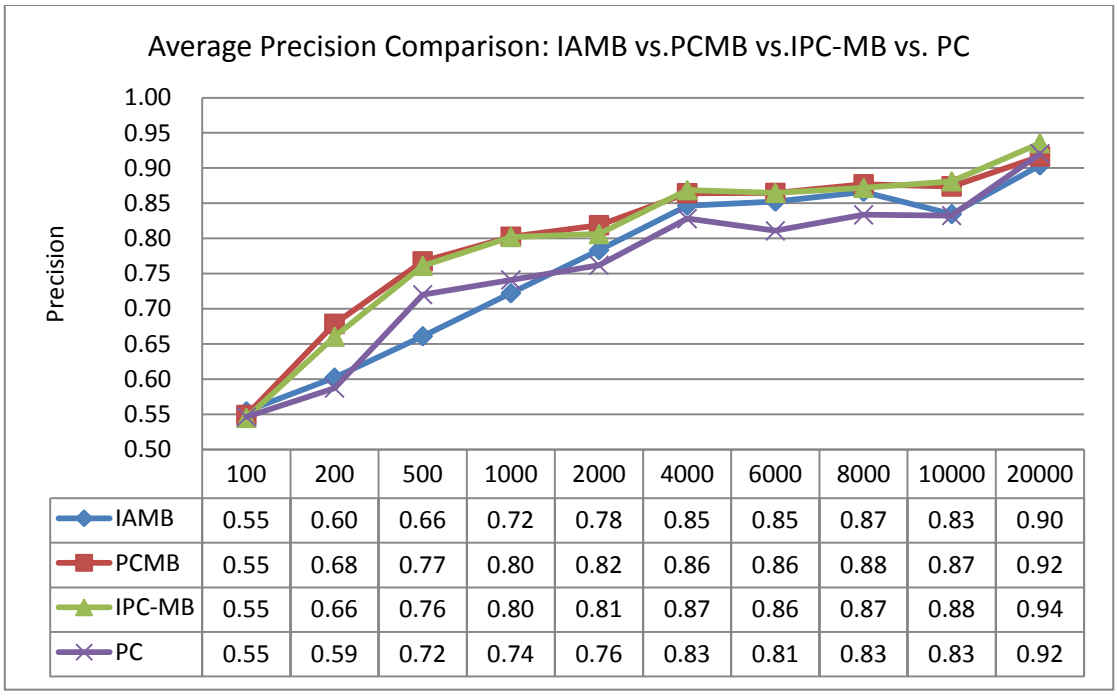
Instances	Simulation Rounds	Algorithm	Precision (Mean ±Std. Dev.)	Recall (Mean ±Std. Dev.)	Distance (Mean ± Std. Dev.)
100	20	IAMB	.55±.08	.51±.09	.72±.10
		PCMB	.55±.11	.49±.17	.76±.15
		IPC-MB	.55±.11	.47±.17	.77±.16
		PC	.55±.14	.60±.26	.71±.13
200	20	IAMB	.60±.09	.72±.09	.53±.11
		PCMB	.68±.10	.57±.13	.61±.11
		IPC-MB	.66±.11	.55±.12	.63±.12
		PC	.59±.14	.62±.22	.66±.10
500	20	IAMB	.66± .06	.77±.05	.45±.08
		PCMB	.77±.10	.65±.10	.48±.13
		IPC-MB	.76±.10	.66±.10	.47±.14
		PC	.72±.12	.64±.10	.52±.13
1000	10	IAMB	.72±.11	.79±.06	.39±.12
		PCMB	.80±.11	.69±.07	.42±.12
		IPC-MB	.80±.12	.73±.09	.38±.14

		PC	.74±11	.70± 09	.45±13
2000	10	IAMB	.78±14	.78±05	.35±13
		PCMB	.82±12	.71±06	.40±11
		IPC-MB	.81±11	.73±01	.38±08
		PC	.76± 09	.69±05	.44±05
4000	10	IAMB	.85±07	.82±09	.26±11
		PCMB	.86±04	.76±11	.31±10
		IPC-MB	.87±02	.76±07	.30±08
		PC	.83±05	.74±08	.35±08
6000	10	IAMB	.85±07	.83±06	.26±10
		PCMB	.86±06	.82±12	.25±13
		IPC-MB	.86±06	.82±09	.25±11
		PC	.81± 04	.81±10	.31±08
8000	10	IAMB	.87±08	.84±08	.24±12
		PCMB	.88±07	.82±11	.24±13
		IPC-MB	.87±04	.82±08	.24±09
		PC	.83± 07	.80±10	.29±11
10000	10	IAMB	.83±08	.83±07	.27±10
		PCMB	.87±06	.81±10	.26±12
		IPC-MB	.88±02	.81±06	.25±06
		PC	.83± 06	.79± 08	.30±07
20000	10	IAMB	.90±06	.92±07	.15±07
		PCMB	.92±08	.93±08	.12±10
		IPC-MB	.94±07	.94±08	.10±10
		PC	.92± 08	.93± 08	.12±10



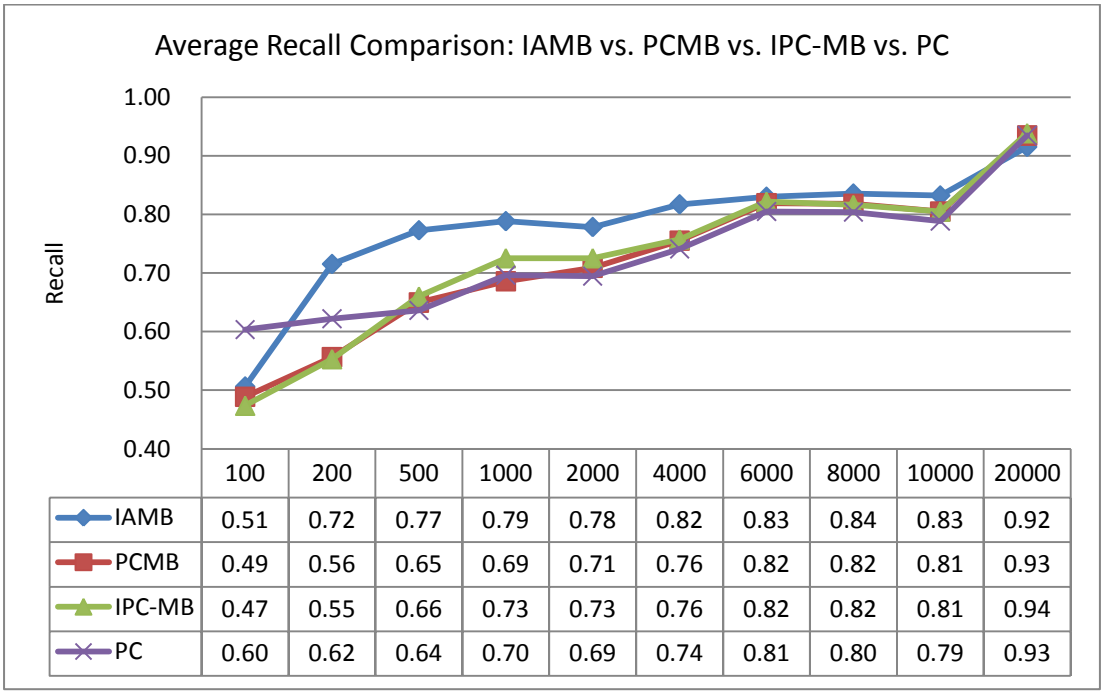
	100	200	500	1000	2000	4000	6000	8000	10000	20000
IAMB	0.72	0.53	0.45	0.39	0.35	0.26	0.26	0.24	0.27	0.15
PCMB	0.76	0.61	0.48	0.42	0.40	0.31	0.25	0.24	0.26	0.12
IPC-MB	0.77	0.63	0.47	0.38	0.38	0.30	0.25	0.24	0.25	0.10
PC	0.71	0.66	0.52	0.45	0.44	0.35	0.31	0.29	0.30	0.12

Figure 4-6: Comparison of distances given different number of instances (0.1K~20K): IAMB vs. PCMB vs. IPC-MB vs. PC (Asia, ε= 0.05, refer to Table 4.2 for more information)



	100	200	500	1000	2000	4000	6000	8000	10000	20000
IAMB	0.55	0.60	0.66	0.72	0.78	0.85	0.85	0.87	0.83	0.90
PCMB	0.55	0.68	0.77	0.80	0.82	0.86	0.86	0.88	0.87	0.92
IPC-MB	0.55	0.66	0.76	0.80	0.81	0.87	0.86	0.87	0.88	0.94
PC	0.55	0.59	0.72	0.74	0.76	0.83	0.81	0.83	0.83	0.92

Figure 4-7: Comparison of precision given different number of instances (0.1K~20K): IAMB vs. PCMB vs. IPC-MB vs. PC (Asia, ε= 0.05, refer to Table 4.2 for more information)



	100	200	500	1000	2000	4000	6000	8000	10000	20000
IAMB	0.51	0.72	0.77	0.79	0.78	0.82	0.83	0.84	0.83	0.92
PCMB	0.49	0.56	0.65	0.69	0.71	0.76	0.82	0.82	0.81	0.93
IPC-MB	0.47	0.55	0.66	0.73	0.73	0.76	0.82	0.82	0.81	0.94
PC	0.60	0.62	0.64	0.70	0.69	0.74	0.81	0.80	0.79	0.93

Figure 4-8: Comparison of recall given different number of instances (0.1K~20K): IAMB vs. PCMB vs. IPC-MB vs. PC (Asia, ε= 0.05, refer to Table 4.2 for more information)

4.4.2 Moderate Network: Alarm

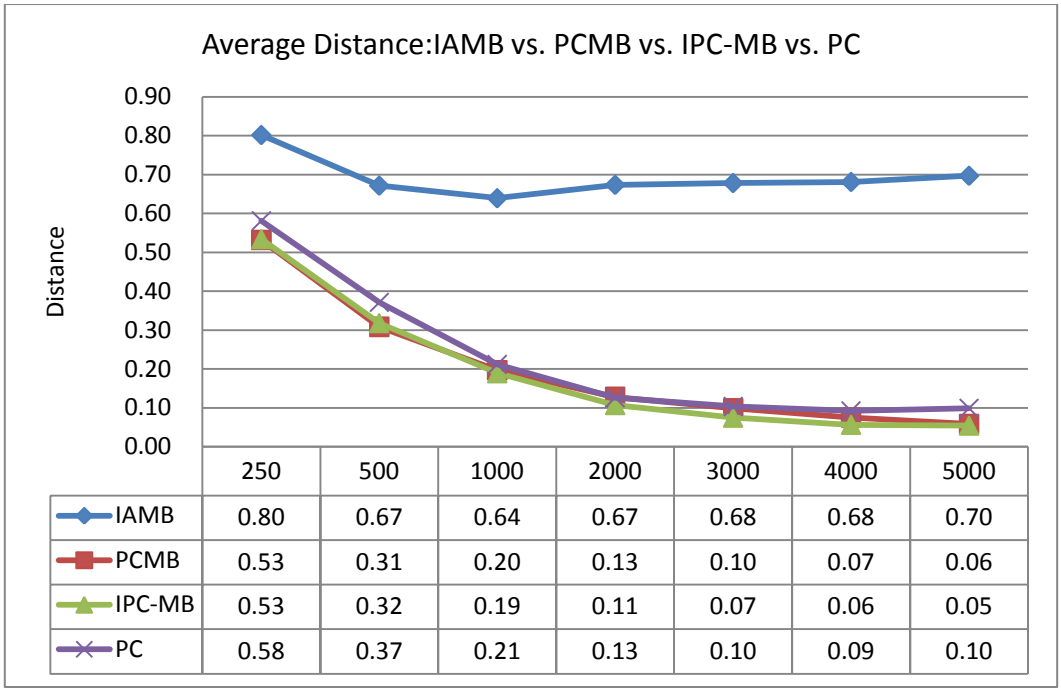
For the Alarm network (Figure 4-2), a 10-folds simulation experiment is conducted considering that it contains many more nodes as compared with Asia, and more stable results are expected. Detailed results are presented in Table 4.3, followed by graphs about the average distance (Figure 4-9), precision (Figure 4-10) and recall (Figure 4-11).

We note that there is a difference between our results and those in [13] on IAMB, given Alarm data. Their accuracy results are close to the IPC-MB results up to 1000 cases. Accuracy (in terms of distance) stands around 0.20 at 2000 and more cases. This discrepancy can be explained that they actually implemented InterIAMB, a variant of IAMB that interleaves the growing and shrinking steps until convergence to improve data efficiency. Hence, the results about IAMB as reported in [13] are, in fact, those of InterIAMB as mentioned in section 4.1 of [13], whereas ours are based on the plain IAMB (Figure 1-3). Another source of discrepancy stems from the fact that

they used 0.01 as the significance value, while we take 0.05. By applying 0.05 to the software package provided by the authors of [13], we empirically observed obviously worse results.

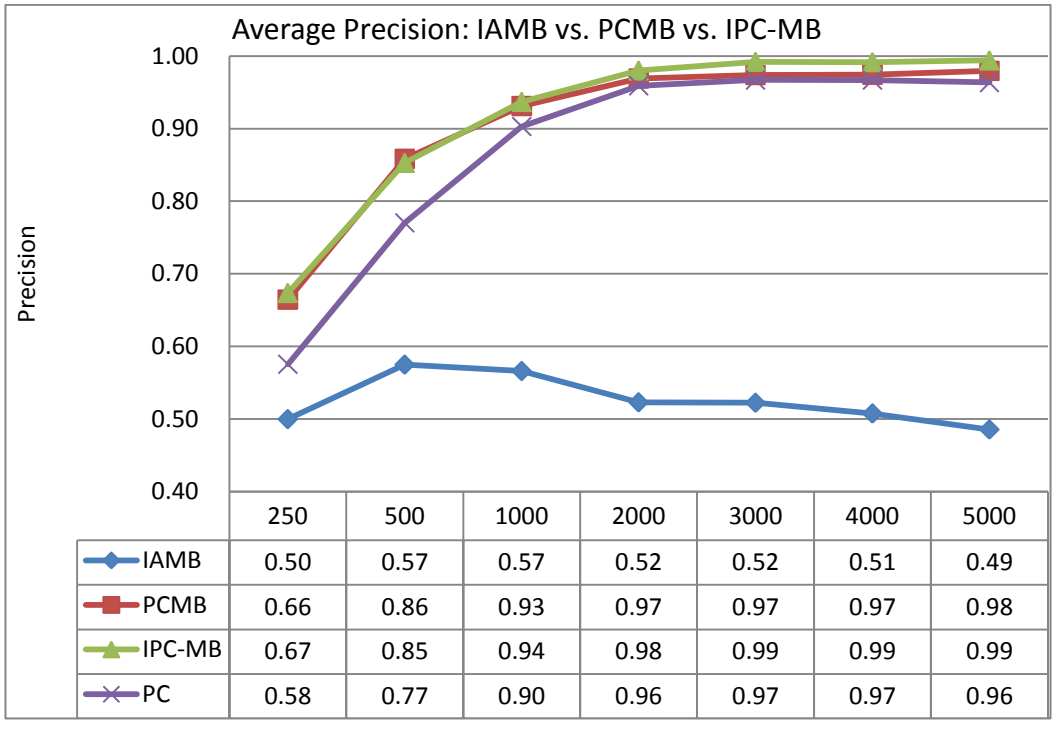
Table 4.3: Accuracy comparison of IAMB, PCMB, IPC-MB and PC over Alarm network.

Instances	**Simulation Rounds**	**Algorithm**	**Precision (Mean ±Std.Dev.)**	**Recall (Mean ±Std.Dev.)**	**Distance (Mean ± Std.Dev.)**
250	10	IAMB	.50±.10	.43±.06	.80±.10
		PCMB	.66±.10	.68±.06	.53±.08
		IPC-MB	.67±.10	.67±.06	.53±.08
		PC	.58±.07	.70±.04	.58±.06
500	10	IAMB	.57±.03	.55±.02	.67±.04
		PCMB	.86±.03	.78±.04	.31±.05
		IPC-MB	.85±.02	.77±.04	.32±.04
		PC	.77±.05	.78±.03	.37±.04
1000	10	IAMB	.57±.02	.60±.02	.64±.02
		PCMB	.93±.02	.84±.02	.20±.03
		IPC-MB	.94±.02	.84±.02	.19±.03
		PC	.90±.03	.85±.03	.21±.04
2000	10	IAMB	.52±.03	.58±.01	.67±.02
		PCMB	.97±.03	.89±.03	.13±.04
		IPC-MB	.98±.02	.90±.03	.11±.04
		PC	.96±.02	.90±.03	.13±.04
3000	10	IAMB	.52±.03	.58±.02	.68±.03
		PCMB	.97±.01	.92±.03	.10±.04
		IPC-MB	.99±.01	.93±.02	.07±.03
		PC	.97±.01	.92±.02	.10±.02
4000	10	IAMB	.51±.03	.59±.02	.68±.03
		PCMB	.97±.02	.94±.03	.07±.04
		IPC-MB	.99±.01	.95±.01	.06±.03
		PC	.97±.01	.94±.02	.09±.03
5000	10	IAMB	.49±.02	.58±.02	.70±.03-
		PCMB	.98±.01	.96±.03	.06±.03
		IPC-MB	.99±.01	.95±.01	.05±.02
		PC	. 96±.02	.94±.01	.10±.03



	250	500	1000	2000	3000	4000	5000
IAMB	0.80	0.67	0.64	0.67	0.68	0.68	0.70
PCMB	0.53	0.31	0.20	0.13	0.10	0.07	0.06
IPC-MB	0.53	0.32	0.19	0.11	0.07	0.06	0.05
PC	0.58	0.37	0.21	0.13	0.10	0.09	0.10

Figure 4-9: Comparison of distances given different number of instances (0.5K~5K): IAMB vs. PCMB vs. IPC-MB vs. PC (Alarm, ε= 0.05, refer to Table 4.3 for more information)



	250	500	1000	2000	3000	4000	5000
IAMB	0.50	0.57	0.57	0.52	0.52	0.51	0.49
PCMB	0.66	0.86	0.93	0.97	0.97	0.97	0.98
IPC-MB	0.67	0.85	0.94	0.98	0.99	0.99	0.99
PC	0.58	0.77	0.90	0.96	0.97	0.97	0.96

Figure 4-10: Comparison of precision given different number of instances (0.5K~5K): IAMB vs. PCMB vs. IPC-MB vs. PC (Alarm, ε= 0.05, refer to Table 4.3 for more information)

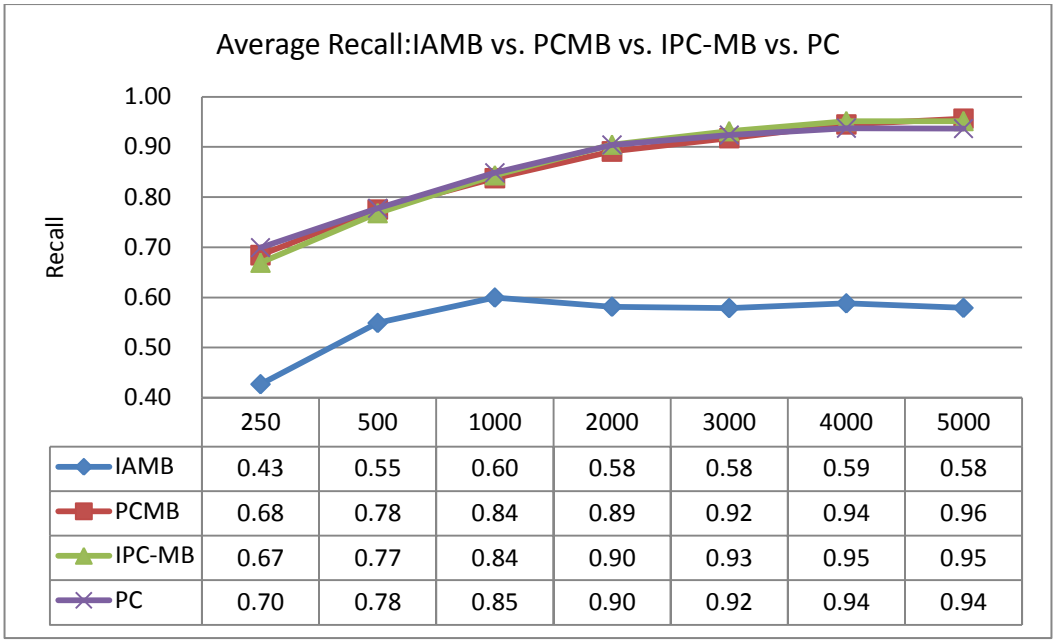


Figure 4-11: Comparison of recall given different number of instances (0.5K~5K): IAMB vs. PCMB vs. IPC-MB vs. PC (Alarm, ε= 0.05, refer to Table 4.3 for more information)

4.4.3 Large Network: Hailfinder and Test152

A 10-folds experiment is also conducted for the Hailfinder and Test152 networks, and the corresponding average accuracy is reported in Table 4.4 and

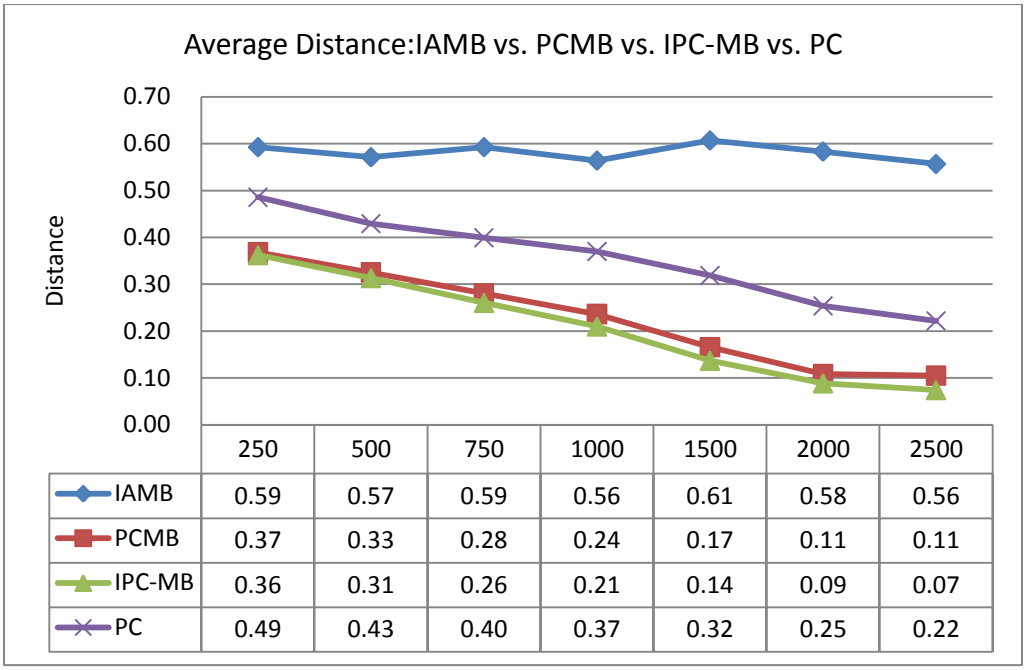
Table 4.5.

Table 4.4: Accuracy comparison of IAMB, PCMB, IPC-MB and PC over Hailfinder network.

Instances	Simulation Rounds	Algorithm	Precision (Mean ±Std. Dev.)	Recall (Mean ±Std. Dev.)	Distance (Mean ± Std. Dev.)
10000	10	IAMB	.36±02	.45±01	.88±02
		PCMB	.71±02	.52±03	.61±03
		IPC-MB	.71±02	.53±02	.60±02
		PC	.70±02	.53±03	.61±02
20000	10	IAMB	.34±01	.45±01	.89±01
		PCMB	.74±03	.56±04	.55±04
		IPC-MB	.73±03	.58±03	.54±05
		PC	.71±04	.57±04	.56±06

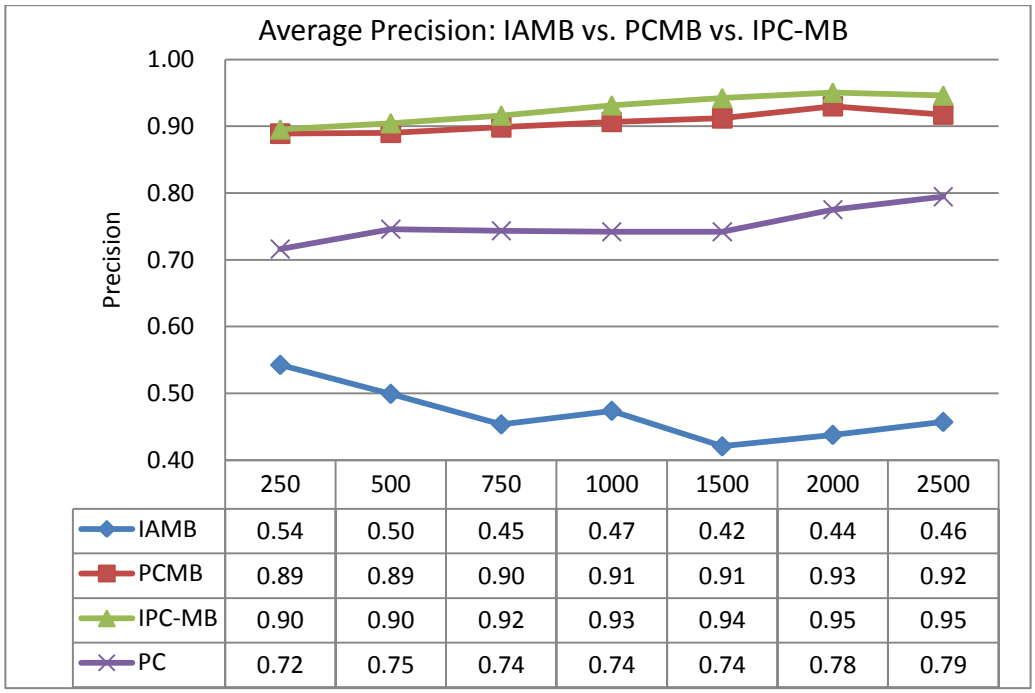
Table 4.5: Accuracy comparison of IAMB, PCMB, IPC-MB and PC over Test152 network.

Instances	Simulation Rounds	Algorithm	Precision (Mean ±Std. Dev.)	Recall (Mean ±Std. Dev.)	Distance (Mean ± Std. Dev.)
250	10	IAMB	.54±01	.74±00	.59±01
		PCMB	.89±02	.71±01	.37±02
		IPC-MB	.90±02	.71±01	.36±01
		PC	.72±03	.71±01	.49±02
500	10	IAMB	.50±01	.81±01	.57±01
		PCMB	.89±01	.76±01	.33±02
		IPC-MB	.90±01	.76±01	.31±01
		PC	.75±03	.76±02	.43±01
750	10	IAMB	.45±01	.86±01	.59±01
		PCMB	.90±03	.80±02	.28±03
		IPC-MB	.92±01	.81±02	.26±02
		PC	.74±04	.80±02	.40±03
1000	10	IAMB	.47±01	.89±01	.56±02
		PCMB	.91±02	.84±02	.24±03
		IPC-MB	.93±02	.85±02	.21±03
		PC	.74±02	.84±02	.37±03
1500	10	IAMB	.42±01	.91±01	.61±01
		PCMB	.91±01	.91±02	.17±03
		IPC-MB	.94±01	.92±02	.14±02
		PC	.74±02	.91±02	.32±03
2000	10	IAMB	.44±01	.93±01	.58±01
		PCMB	.93±01	.96±02	.11±02
		IPC-MB	.95±01	.96±02	.09±02
		PC	.78±02	.96±01	.25±02
2500	10	IAMB	.46±01	.96±01	.56±01
		PCMB	.92±02	.97±01	.11±02
		IPC-MB	.95±01	.98±01	.07±01
		PC	.79±02	.98±01	.22±01



	250	500	750	1000	1500	2000	2500
IAMB	0.59	0.57	0.59	0.56	0.61	0.58	0.56
PCMB	0.37	0.33	0.28	0.24	0.17	0.11	0.11
IPC-MB	0.36	0.31	0.26	0.21	0.14	0.09	0.07
PC	0.49	0.43	0.40	0.37	0.32	0.25	0.22

Figure 4-12: Comparison of distances given different number of instances (0.25K~2.5K): IAMB vs. PCMB vs. IPC-MB vs. PC (Test152, ε= 0.05, refer to Table 4.5 for more information)



	250	500	750	1000	1500	2000	2500
IAMB	0.54	0.50	0.45	0.47	0.42	0.44	0.46
PCMB	0.89	0.89	0.90	0.91	0.91	0.93	0.92
IPC-MB	0.90	0.90	0.92	0.93	0.94	0.95	0.95
PC	0.72	0.75	0.74	0.74	0.74	0.78	0.79

Figure 4-13: Comparison of precision given different number of instances (0.25K~2.5K): IAMB vs. PCMB vs. IPC-MB vs. PC (Test152, ε= 0.05, refer to Table 4.5 for more information)

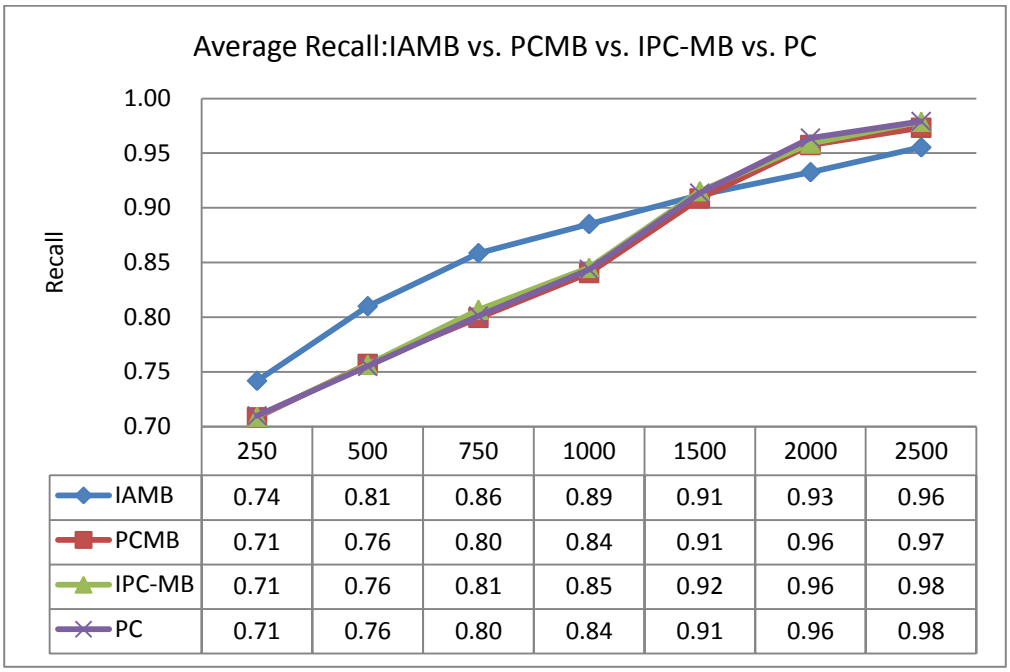


Figure 4-14: Comparison of recall given different number of instances (0.25K~2.5K): IAMB vs. PCMB vs. IPC-MB vs. PC (Test152, ε= 0.05, refer to Table 4.5 for more information).

4.4.4 Polytree Network: PolyAlarm (Derived from Alarm)

A 10-folds experiment for the PolyAlarm network is reported in Figure 4-3. Detailed results are presented in Table 4.6, followed by graphs about the average distance (Figure 4-15), precision (Figure 4-16) and recall (Figure 4-17).

Table 4.6: Accuracy comparison of IAMB, PCMB, IPC-MB and PC over polytree version Alarm network.

Instances	Simulation Rounds	Algorithm	Precision (Mean ±Std. Dev.)	Recall (Mean ±Std. Dev.)	Distance (Mean ± Std. Dev.)
500	10	IAMB	.64±.03	.71±.03	.53±.04
		PCMB	.84±.05	.75±.04	.33±.07

		IPC-MB	.85±05	.74±04	.33±07
		PC	.76±07	.72±05	.43±08
1000	10	IAMB	.70±03	.84±02	.40±04
		PCMB	.91±02	.86±01	.19±02
		IPC-MB	.91±03	.85±02	.20±04
		PC	.81±04	.80±02	.34±06
2000	10	IAMB	.65±02	.89±01	.42±02
		PCMB	.93±02	.89±02	.14±02
		IPC-MB	.93±01	.90±03	.13±04
		PC	.83±03	.83±02	.29±04
3000	10	IAMB	.65±03	.89±02	.41±02
		PCMB	.91±02	.92±02	.13±05
		IPC-MB	.92±02	.91±03	.13±04
		PC	.84±03	.86±01	.26±03
4000	10	IAMB	.62±03	.92±02	.43±04
		PCMB	.93±03	.92±02	.13±05
		IPC-MB	.94±02	.92±02	.12±03
		PC	.86±03	.87±03	.23±04
5000	10	IAMB	.61±04	.92±02	.43±05
		PCMB	.93±03	.93±02	.11±04
		IPC-MB	.94±02	.92±02	.11±02
		PC	.87±03	.89±03	.20±04

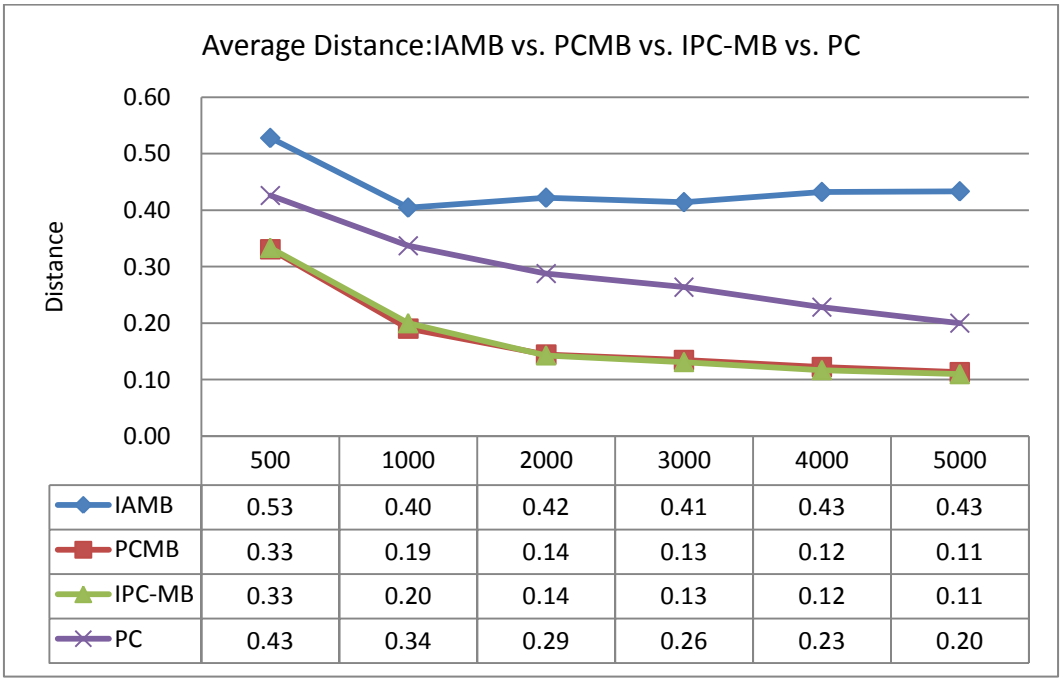
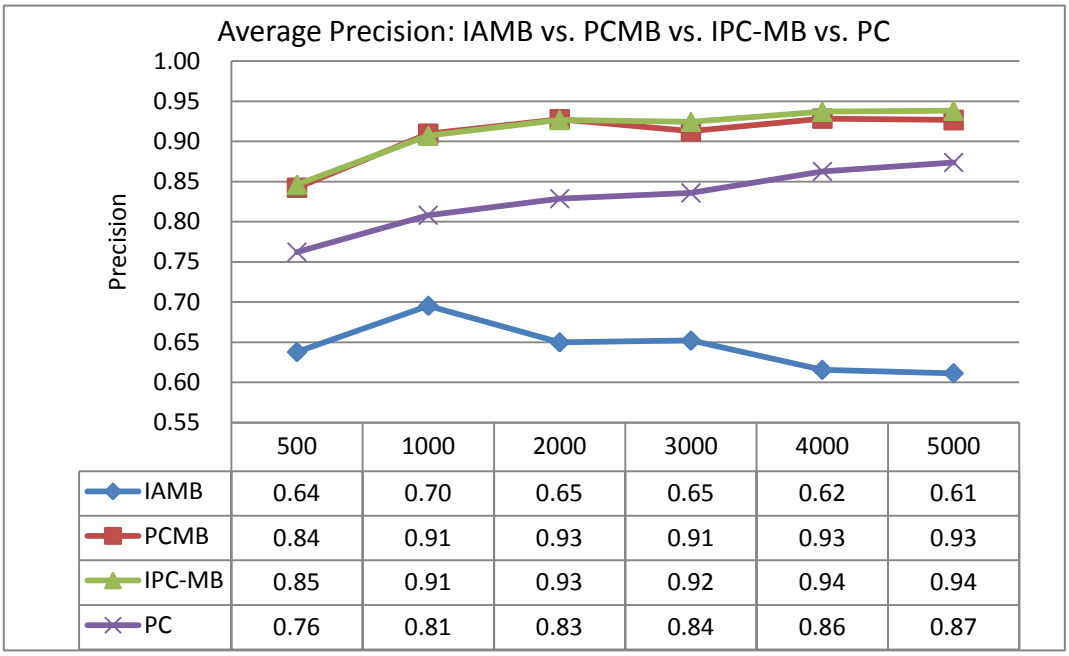
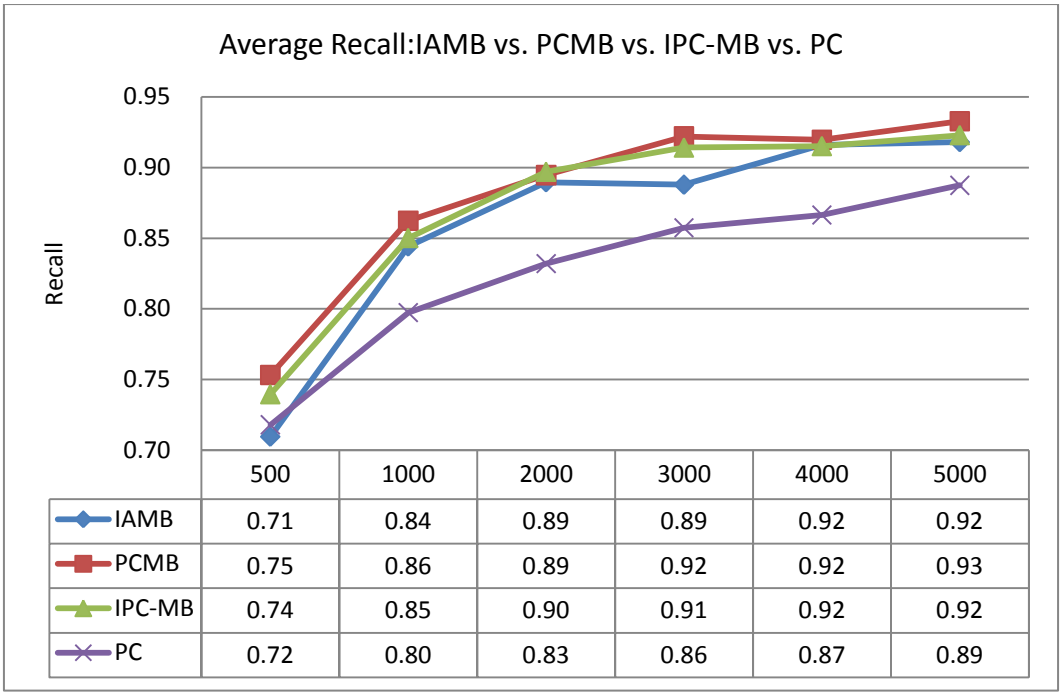


Figure 4-15: Comparison of distances given different number of instances (0.5K~5K): IAMB vs. PCMB vs. IPC-MB vs. PC (PolyAlarm, ε= 0.05, refer to Table 4.6 for more information)



	500	1000	2000	3000	4000	5000
IAMB	0.64	0.70	0.65	0.65	0.62	0.61
PCMB	0.84	0.91	0.93	0.91	0.93	0.93
IPC-MB	0.85	0.91	0.93	0.92	0.94	0.94
PC	0.76	0.81	0.83	0.84	0.86	0.87

Figure 4-16: Comparison of precision given different number of instances (0.5K~5K): IAMB vs. PCMB vs. IPC-MB vs. PC (PolyAlarm, ε= 0.05, refer to Table 4.6 for more information)



	500	1000	2000	3000	4000	5000
IAMB	0.71	0.84	0.89	0.89	0.92	0.92
PCMB	0.75	0.86	0.89	0.92	0.92	0.93
IPC-MB	0.74	0.85	0.90	0.91	0.92	0.92
PC	0.72	0.80	0.83	0.86	0.87	0.89

Figure 4-17: Comparison of recall given different number of instances (0.5K~5K): IAMB vs. PCMB vs. IPC-MB vs. PC (PolyAlarm, ε= 0.05, refer to Table 4.6 for more information).

4.4.5 Conclusion

Our experiments with different size of samples over five different scale of networks indicate that:

- As expected, the accuracy of PCMB, IPC-MB and PC increases when more observations become available (with decreasing distance in Figure 4-6, Figure 4-9, Figure 4-12, and Figure 4-15). However, though it is believed that IAMB will also produce perfect results given enough data, it appears the accuracy of IAMB flattens quickly given more observations, e.g. in the problems of Alarm (Figure 4-9), Test152 (Figure 4-12) and PolyAlarm (Figure 4-15). This may be explained by the fact that more false positives are added in the growing phase (see more discussion in Section 4.6 and 5.4);
- The underlying topology, or the problem itself, greatly influences the performance of all algorithms. Given the same amount of observations, we observe quite different accuracy performances in different problems;

- The algorithms may produce worse results in problems with fewer features, but more observations. For example, with 20,000 instances, the accuracy reached in the other four problems is much lower than that achieved with Test152 in which only 2,500 instances are given. Therefore, the underlying topology, together with the number of states of nodes (or variables), determines the actual complexity of problems;
- PCMB and IPC-MB demonstrate no obvious gain over IAMB given very small problem like Asia (refer to Figure 4-6). However, the relative advantage becomes quite attractive given larger problems (Figure 4-9, Figure 4-12 and Figure 4-15). For example, with Alarm network, PCMB and IPC-MB have distance less than 0.50 given 500 instances, but IAMB couldn't reach this level even with as many as 5,000 instances;
- With more observations being fed with, PCMB and IPC-MB have much faster increase in accuracy than IAMB;
- PCMB has close performance to IPC-MB, in term of both precision and recall; IPC-MB performs slightly better than PCMB;
- IPC-MB never loses to PC, and has obvious better accuracy in Test152 (Figure 4-12) and PolyAlarm (Figure 4-15);
- Given more data, both precision and recall increase for PCMB and IPC-MB algorithms. However, precision always is higher than recall as observed in our experiments, before enough information becomes available for them to reach a balance. Figure 4-18 illustrates this difference given IPC-MB as example. For PCMB, this reflects that its strict selection of true positives is effective; and for IPC-MB, it confirms that its strategy of removing as many as possible false positives also works quite well;

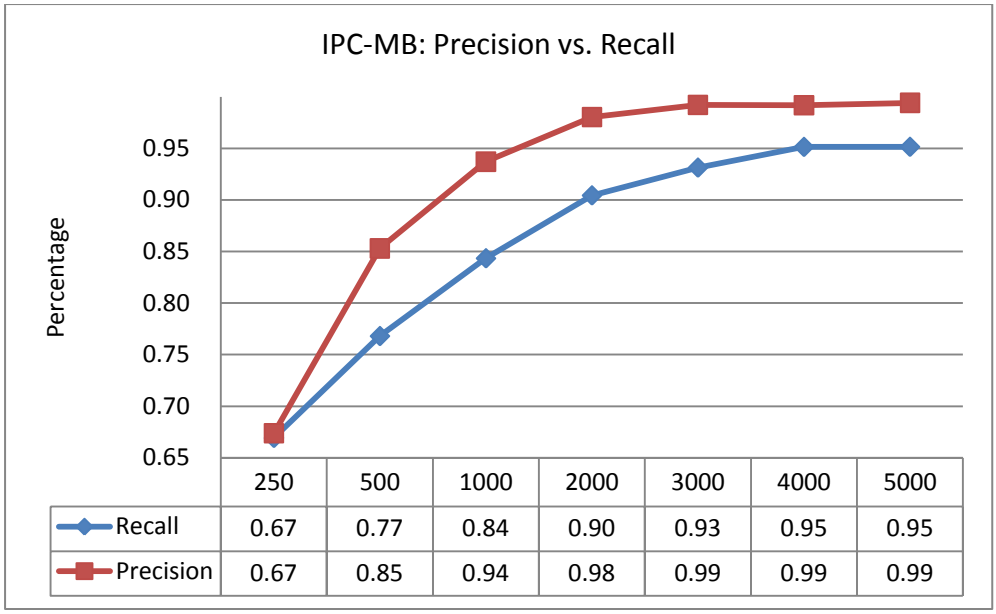


Figure 4-18: Comparison of IPC-MB's Precision and Recall (Based on experiments with Alarm, ε= 0.05, refer to Table 4.3 for more information)

In conclusion, although IAMB, PCMB and IPC-MB algorithms are all proved correct, and they are believed to produce the perfect result if enough data is available, their relative accuracy performance is different given limited observations. Obvious difference is observed between IAMB and PCMB/IPC-MB.

IPC-MB has slightly higher accuracy performance than PCMB, and it beats PC with obvious advantage in some cases.

4.5 Time Efficiency

In Chapter 3, we have analyzed the time complexity of IPC-MB in terms of number of CI tests. Here, one more measure is introduced, that is number of data passes, where a data pass consists in scanning the whole training data for one time. In practice, to save the memory, we generally do not cache all contingency tables; in fact, it is impossible to do so given the exponentially growing number of possible subsets (or combinations) of features. Hence, a more practical way is to cache only what are known as necessary for our immediate decision making purpose. For example, in IPC-MB, we only cache the contingency tables given the known T, X and conditioning set of cardinality *cutSetSize*, and this is reasonable since not only the *cutSetSize* will change (increase with 1) in next iteration, but also the $\mathbf{PC}_T^C$ due to the possible deletion of false positives.

Therefore, we only allocate memory to cache $\mathbf{S} \subseteq \mathbf{PC}_T^C$ with $|\mathbf{S}| = cutSetSize$, and release all memory allocation at the end of this iteration, which is effective to reduce the consumption of space. However, this requires to re-scan the whole data file with additional time upon entering next iteration, e.g. with increased *cutSetSize* and possibly modified $\mathbf{PC}_T^C$. With very large training data, scanning the whole data may be quite time consuming a job since we may need to visit the disk, or even network, for many times.

To make different algorithm comparable, we treat the collection of contingency tables in a fair manner in our implementation, i.e. one data pass is consumed to collect all statistics expected in the current active session or loop. In IAMB, an additional data pass is needed to re-construct related contingency tables after adding or removing one variable. While in PCMB, three data passes are needed in each iteration of the searching loop within *GetPCD*, corresponding to the three steps involved, i.e. removing negatives, adding best candidate and removing false positives respectively.

4.5.1 Small Network: Asia

In Table 4.7, the "# Data Passes" of IAMB/PCMB/IPC-MB refers to the average number of data passes we need to induce the corresponding Markov blanket of all the 8 nodes of Asia BN (It is obtained by dividing the total number of data passes with the number of nodes, to reflect the average complexity of IAMB, PCMB and IPC-MB), while the "# Data Passes" of PC is just the number of data passes happening to induce the whole network. "# CI test" is defined similarly. Generally, the larger are these two numbers, the slower is the algorithm.

Table 4.7: Time complexity comparison of IAMB, PCMB, IPC-MB and PC over Asia network.

Instances	Simulation Rounds	Algorithm	# Data Passes (Mean ±Std. Dev.)	# CI Tests (Mean ±Std. Dev.)
100	20	IAMB	5 ±1	25 ±3
		PCMB	80 ±87	2006 ±3673
		IPC-MB	10 ±7	188 ±288
		PC	26 ±9	213 ±267
200	20	IAMB	4 ±0	22 ±2
		PCMB	64 ±50	1044 ±1636
		IPC-MB	8 ±4	110 ±136
		PC	25 ±8	171 ±179
500	20	IAMB	5 ±0	23 ±1
		PCMB	48 ±12	316 ±130

		IPC-MB	8 ±2	63 ±13
		PC	24 ±3	111 ±18
1000	10	IAMB	5 ±0	23 ±1
		PCMB	49 ±13	367 ±122
		IPC-MB	8 ±1	70 ±11
		PC	24 ±3	120 ±15
2000	10	IAMB	5 ±0	23 ±2
		PCMB	52 ±17	433 ±213
		IPC-MB	9 ±2	77 ±18
		PC	24 ±3	131 ±23
4000	10	IAMB	5 ±0	23 ±1
		PCMB	50 ±7	436 ±84
		IPC-MB	8 ±1	84 ±9
		PC	26 ±4	139 ±10
6000	10	IAMB	5 ±0	23 ±1
		PCMB	55 ±10	486 ±104
		IPC-MB	9 ±1	91 ±16
		PC	27 ±4	147 ±19
8000	10	IAMB	5 ±0	23 ±1
		PCMB	55 ±9	482 ±98
		IPC-MB	9 ±1	90 ±15
		PC	28 ±4	147 ±19
10000	10	IAMB	5 ±0	24 ±1
		PCMB	57 ±8	493 ±75
		IPC-MB	9 ±1	91 ±12
		PC	27 ±4	150 ±17
20000	10	IAMB	5 ±0	25 ±1
		PCMB	66 ±11	583 ±109
		IPC-MB	10 ±1	99 ±13
		PC	31 ±3	155 ±14

4.5.2 Moderate Network: Alarm

Table 4.8: Time complexity comparison of IAMB, PCMB, IPC-MB and PC over ALARM network.

Instances	**Simulation Rounds**	**Algorithm**	**# Data Passes (mean ±Std. Dev.)**	**# CI Tests (mean ±Std. Dev.)**
250	10	IAMB	4 ±0	93 ±8
		PCMB	261 ±22	5464 ±539
		IPC-MB	12 ±1	562 ±30
		PC	303 ±19	2330 ±92
500	10	IAMB	5 ±0	116 ±2
		PCMB	160 ±11	4638 ±374

		IPC-MB	12 ±1	561 ±31
		PC	220 ±16	2736 ±82
1000	10	IAMB	5 ±0	140 ±2
		PCMB	154 ±5	6047 ±385
		IPC-MB	12 ±0	637 ±37
		PC	191 ±12	3528 ±121
2000	10	IAMB	6 ±0	162 ±2
		PCMB	175 ±7	8804 ±532
		IPC-MB	13 ±0	736 ±37
		PC	188 ±12	3528 ±121
3000	10	IAMB	6 ±0	179 ±3
		PCMB	204 ±8	12329 ±817
		IPC-MB	13 ±0	798 ±53
		PC	200 ±19	3717 ±166
4000	10	IAMB	7 ±0	187 ±4
		PCMB	218 ±6	16007 ±1326
		IPC-MB	14 ±0	849 ±48
		PC	211 ±18	3902 ±122
5000	10	IAMB	7 ±0	197 ±3
		PCMB	231 ±6	17704 ±1189
		IPC-MB	14 ±0	876 ±31
		PC	215 ±16	3956 ±80

4.5.3 Large Network: Hailfinder and Test152

Table 4.9: Time complexity comparison of IAMB, PCMB, IPC-MB over Hailfinder network (ε= 0.05).

Instances	Simulation Rounds	Algorithm	# Data Passes (mean ±Std. Dev.)	# CI Tests (mean ±Std. Dev.)
10000	10	IAMB	6 ±0	270 ±1
		PCMB	120 ±10	8186 ±1049
		IPC-MB	9 ±0	736 ±56
		PC	283 ±25	6489 ±161
20000	10	IAMB	7 ±0	299 ±2
		PCMB	136 ±10	14538 ±1617
		IPC-MB	10 ±0	1000 ±101
		PC	284 ±29	7515 ±301

Table 4.10: Time complexity comparison of IAMB, PCMB, IPC-MB over Test152 network (ε= 0.05).

Instances	Simulation Rounds	Algorithm	# Data Passes (mean ±Std. Dev.)	# CI Tests (mean ±Std. Dev.)

250	10	IAMB	5 ±0	602 ±0
		PCMB	89 ±4	3154 ±168
		IPC-MB	11 ±0	780 ±29
		PC	608 ±3	17947 ±351
500	10	IAMB	6 ±0	750 ±1
		PCMB	101 ±3	3757 ±148
		IPC-MB	12 ±0	924 ±28
		PC	669 ±78	19803 ±392
750	10	IAMB	7 ±0	896 ±2
		PCMB	111 ±6	4284 ±239
		IPC-MB	13 ±1	1055 ±52
		PC	684 ±80	21429 ±582
1000	10	IAMB	7 ±0	896 ±2
		PCMB	119 ±4	4685 ±183
		IPC-MB	14 ±0	1147 ±32
		PC	684 ±80	22732 ±426
1500	10	IAMB	8 ±0	1042 ±1
		PCMB	134 ±3	5384 ±145
		IPC-MB	15 ±0	1316 ±35
		PC	714 ±73	24865 ±415
2000	10	IAMB	8 ±0	1041 ±2
		PCMB	148 ±3	5928 ±174
		IPC-MB	15 ±0	1432 ±46
		PC	684 ±80	26173 ±593
2500	10	IAMB	8 ±0	1042 ±2
		PCMB	161 ±3	6444 ±142
		IPC-MB	16 ±0	1532 ±44
		PC	730 ±96	27512 ±614

4.5.4 Polytree Network: PolyAlarm(Derived)

Table 4.11: Time complexity comparison of IAMB, PCMB, IPC-MB and PC over PolyAlarm network.

Instances	**Simulation Rounds**	**Algorithm**	**# Data Passes (mean ±Std. Dev.)**	**# CI Tests (mean ±Std. Dev.)**
500	10	IAMB	4 ±0	106 ±3
		PCMB	47 ±3	584 ±48
		IPC-MB	7 ±0	143 ±8
		PC	117 ±16	1061 ±48
1000	10	IAMB	5 ±0	126 ±3
		PCMB	54 ±4	715 ±84
		IPC-MB	8 ±0	164 ±8
		PC	140 ±26	1145 ±42

2000	10	IAMB	5 ±0	147 ±2
		PCMB	59 ±2	837 ±57
		IPC-MB	9 ±0	179 ±6
		PC	158 ±24	1223 ±35
3000	10	IAMB	6 ±0	155 ±4
		PCMB	68 ±5	1002 ±78
		IPC-MB	9 ±0	190 ±7
		PC	174 ±15	1265 ±39
4000	10	IAMB	6 ±0	165 ±3
		PCMB	68 ±5	1002 ±78
		IPC-MB	9 ±0	195 ±8
		PC	176 ±11	1292 ±41
5000	10	IAMB	6 ±0	171 ±4
		PCMB	70 ±4	1067 ±41
		IPC-MB	9 ±1	196 ±11
		PC	181 ±2	1308 ±56

4.5.5 Conclusion

Our experiments with different size of samples over five different problems indicate that:

- IAMB has the fastest speed among the four algorithms and IPC-MB is second; PCMB and PC are slower than the other two, and PCMB is the slowest one among the three local search algorithms;
- The underlying topology, i.e. the problem itself, influences the actual performance of all algorithms greatly, especially on PCMB. For example, PCMB may need 347.5% more CI tests than PC in Alarm problem (Table 4.8), but 76.6% less in Test152 problem (Table 4.10). In contrast, the topology has much smaller influence on IAMB, or in other words, IAMB is "blind" to the topology, which confirms the fact that IAMB and its variants don't consider topology in the search;
- Given the same number of features, the actual connectivity influences the actual cost for all algorithms. Generally saying, they cost less on parse networks (Table 4.13);
- Compared with the global search by PC, IPC-MB saves a lot of data passes and CI tests in all experiments (Table 4.12), which reflects the advantage of local search. For example, given the Test152 problem, IPC-MB requires 94.4% fewer times of CI tests than PC, when there are 2,500 instances available for learning;

Table 4.12: Time complexity comparison of between IAMB/PCMB/IPC-MB and PC. The comparison is based on the average measures of 20K-Asia experiment, 5K-PolyAlarm experiment, 5K-Alarm, 20K-Hailfinder and 2.5K-Test152 experiments respectively. In the table ↓ x% means that x% reduction is achieved compared with PC algorithm; ↑ x%, in contrast, indicates additional x% cost relative to that of PC algorithm.

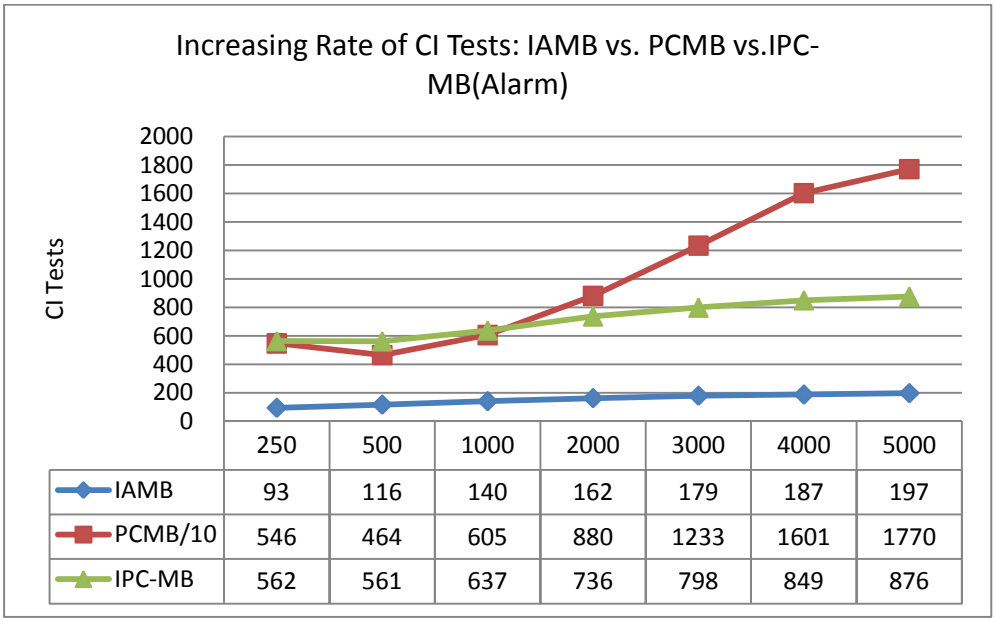
	Problem	**PC**	
		# Data Passes	# CI Tests
IAMB	Asia	↓ 83.9%	↓ 83.9%
	PolyAlarm	↓ 96.7%	↓ 86.9%
	Alarm	↓ 96.7%	↓ 95.0%
	Hailfinder	↓ 97.9%	↓ 96.4%
	Test152	↓ 98.9%	↓ 96.2%
PCMB	Asia	↑ 112.9%	↑ 276.1%
	PolyAlarm	↓ 61.3%	↓ 18.4%
	Alarm	↑ 7.4%	↑ 347.5%
	Hailfinder	↓ 52.1%	↑ 93.4%
	Test152	↓ 77.9%	↓ 76.6%
IPC-MB	Asia	↓ 67.7%	↓ 36.1%
	PolyAlarm	↓ 95.0%	↓ 86.5%
	Alarm	↓ 92.6%	↓ 77.39
	Hailfinder	↓ 96.5%	↓ 86.7

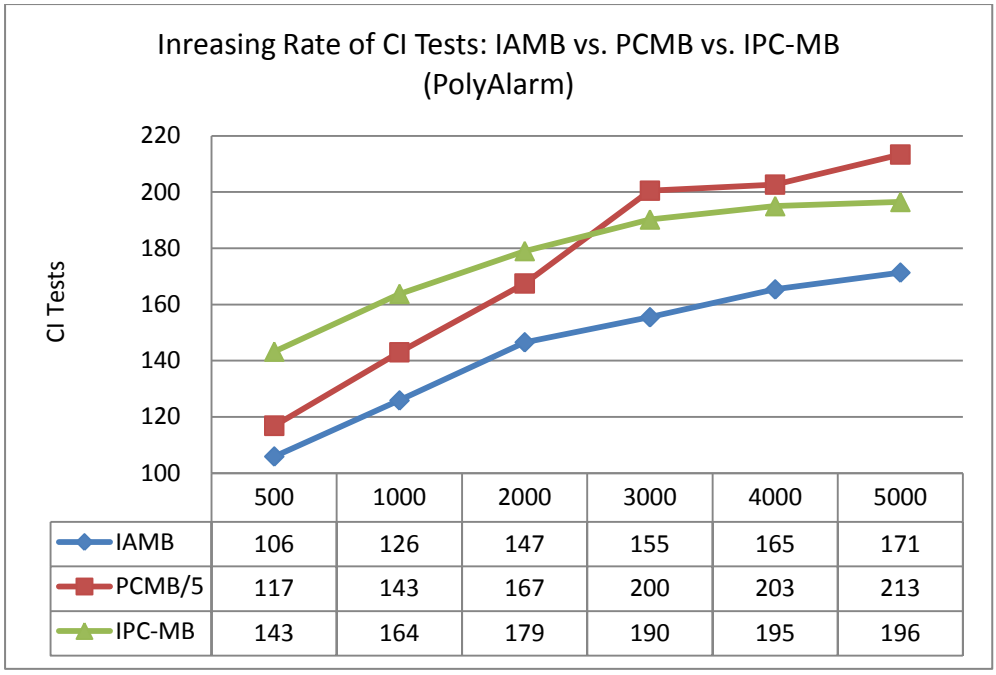
	Test152	↓ 97.8%	↓ 94.4

Table 4.13: Time complexity comparison of IAMB/PCMB/IPC-MB given example networks with same number of nodes but different density of connectivity. All are measured in experiments with 5,000 instances.

		Alarm	
	Algorithm	# Data Passes	# CI Tests
PolyAlarm	IAMB	↓ 14.3%	↓ 13.2%
	PCMB	↓ 69.7%	↓ 94.0%
	IPC-MB	↓ 35.7%	↓ 77.6%

- Given the special network, polytree, the difference between IAMB and IPC-MB becomes very small;
- Though considered a local search, PCMB's cost is similar to PC which conducts global search to induce the whole network;
- IPC-MB is much faster than PCMB, over 75% reduction on CI tests and more than 90% reduction on data passes, in all experiments;
- PCMB has a much higher increasing rate of data passes and CI tests than IPC-MB and IAMB (Figure 4-19). The difference observed in complex problem (Alarm) is more obvious than simpler problem (PolyAlarm). It is easy to understand since the algorithms converge more quickly in simpler problems.





	500	1000	2000	3000	4000	5000
IAMB	106	126	147	155	165	171
PCMB/5	117	143	167	200	203	213
IPC-MB	143	164	179	190	195	196

Figure 4-19: Comparison of increasing rate of CI tests given Alarm and PolyAlarm networks: IAMB vs. PCMB vs. IPC-MB.

4.6 Data Efficiency

Data efficiency can be measured in two dimensions, the relative accuracy given the same amount of training instances, and the actual cardinality of conditioning set as involved in the CI tests. The first measure is indirect, while the second one is direct.

4.6.1 Relative Accuracy

From the study of Section 4.4, it is observed that although IPC-MB has no gain over IAMB given the small problem Asia, it obviously exceeds IAMB given larger and more complex problems, like Alarm, PolyAlam, Hailfinder and Test152. For example, given only 500 instances in Alarm problem, the average distance of IPC-MB is 0.32, while it is 0.67 for IAMB. Besides, given more instances, the accuracy rate reached by IPC-MB increases faster than IAMB, which reflects further that IPC-MB is able to make better use of data to infer more information than IAMB. PCMB and PC perform much better than IAMB too, but slightly poorer than IPC-MB, which will be explained soon in 4.6.2.

With IAMB, we even observe a decrease in accuracy given more observations, e.g. with Alarm (Figure 4-9), Hailfinder (Table 4.9), Test152 (Figure 4-12) and PolyAlarm problems (Figure 4-15). As we mentioned in 4.4.5, this is not our implementation mistake, but it is determined by the nature of IAMB – though more search can be conducted given more instances in IAMB, it adds more false positives in the growing phase but not able to remove them in the shrinking phase.

PCMB is observed to have the similar accuracy performance as IPC-MB, so it is indeed more data efficient than IAMB too as declared in [13].

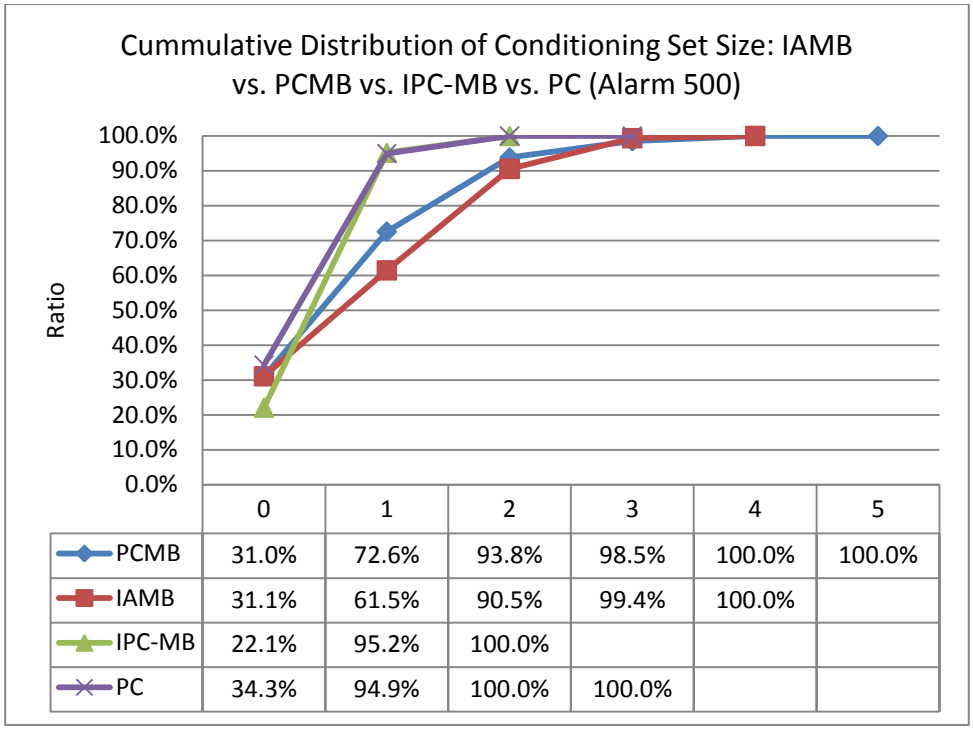
4.6.2 Distribution of Conditioning Set Size

The relative accuracy of IAMB/PCMB/IPC-MB given the same amount of observations is one important, but indirect, measure to reflect the distinction resulted from different data efficiency. In this section, we study the problem in a direct manner by measuring the distribution of conditioning set size of three algorithms, which is believed helpful for us better understand what happens behind the scene.

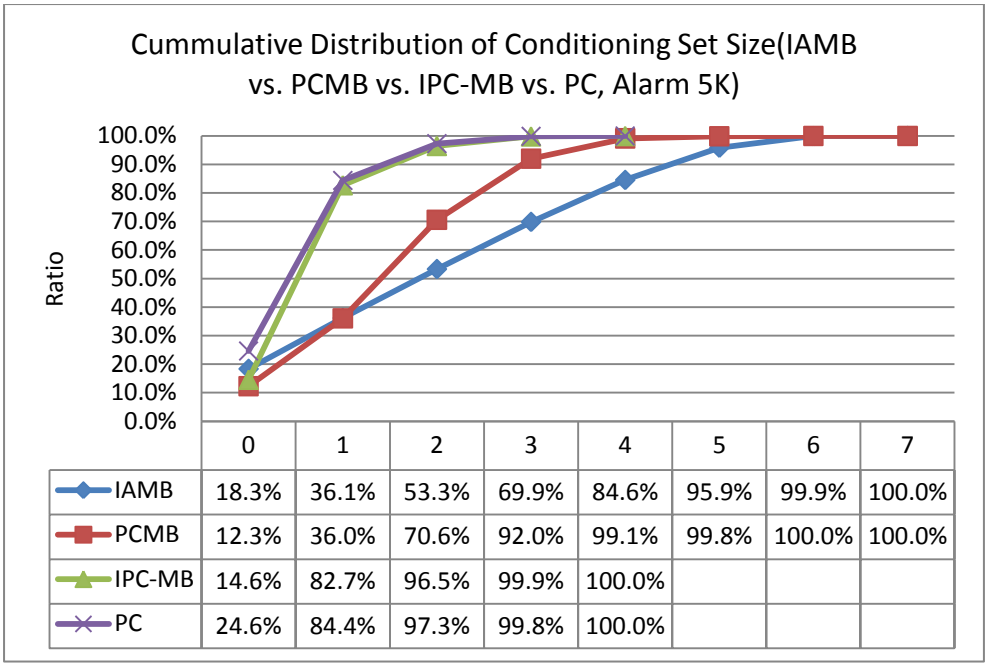
Two example distributions are illustrated in Figure 4-20, and both are based on experiments with Alarm data. We summarize the number of CI tests with conditioning set of cardinality X, and then normalize them with the total number of CI tests as involved in the search to get the relative frequencies. The upper graph in Figure 4-20 is measured given the data collected in the Alarm experiments with 500 instances, and the bottom one is about experiments with 5,000 instances. This permits us to observe the relative distribution of the conditioning set size about IAMB, PCMB, IPC-MB and PC, given “small” and “large” data sets respectively.

From Figure 4-20, we see that given 500 instances, the largest conditioning set is of size five (variable) (found in PCMB); with more instances, for example 5,000, it increases to 7 (found in IAMB and PCMB). The increased largest conditioning set indicates that more searches can be done in IAMB and PCMB. However, we didn’t see any gain in accuracy on IAMB algorithm (Figure 4-9), while PCMB achieves great progress with distance decreasing from 0.31 to 0.06. In contrast, the largest conditioning set is of size 2 and 4 respectively for IPC-MB, given 500 and 5,000 instances.

IPC-MB performs much better than IAMB, and achieves slightly higher accuracy than PCMB, which seems can resort to the fact that most of its CI tests involve fewer number of variables (Figure 4-20, Figure 4-21, Figure 4-22). For example, given 5,000 instances in Alarm problem, 96.6% CI tests have no more than two variables in their conditioning set, in IPC-MB; this number is 70.6% for PCMB, and 53.3% for IAMB. Actually, in all five experiments, we observe that over 90% of CI tests involved in IPC-MB have two or fewer variables in the conditioning set. This explains why PCMB and IPC-MB performs much better than IAMB, in a different light. Besides, we do observe a little more gain by IPC-MB over PCMB in Test152 problem. Though there is no obvious gain is observed by IPC-MB over PCMB, it is believed that more trustable outcomes are expected on IPC-MB over PCMB in applications.

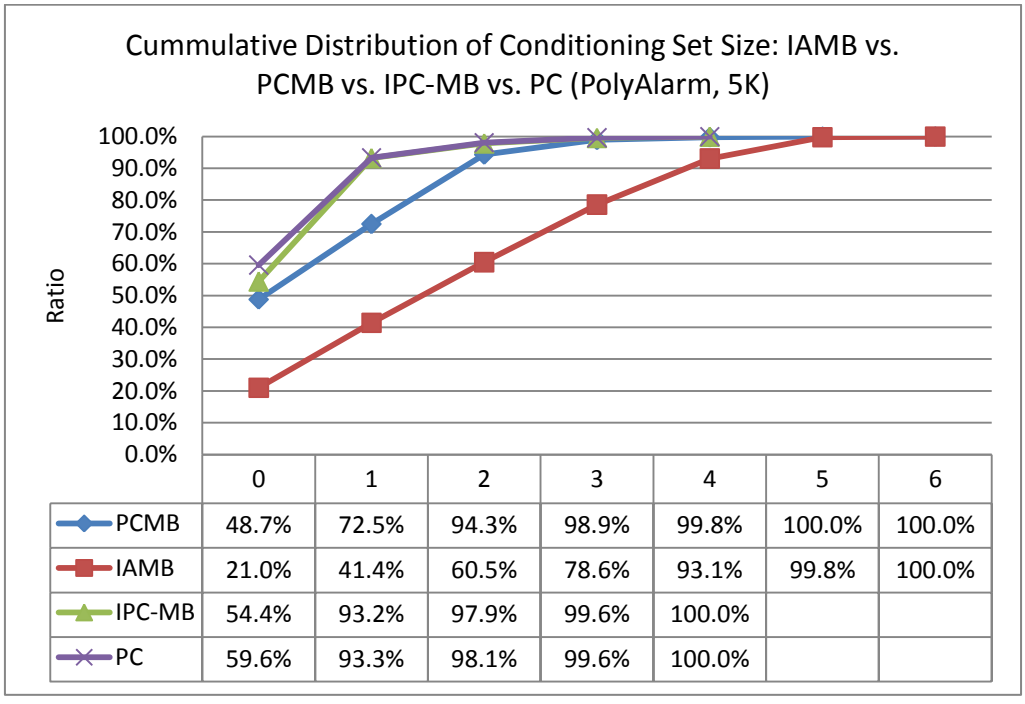


	0	1	2	3	4	5
PCMB	31.0%	72.6%	93.8%	98.5%	100.0%	100.0%
IAMB	31.1%	61.5%	90.5%	99.4%	100.0%	
IPC-MB	22.1%	95.2%	100.0%			
PC	34.3%	94.9%	100.0%	100.0%		



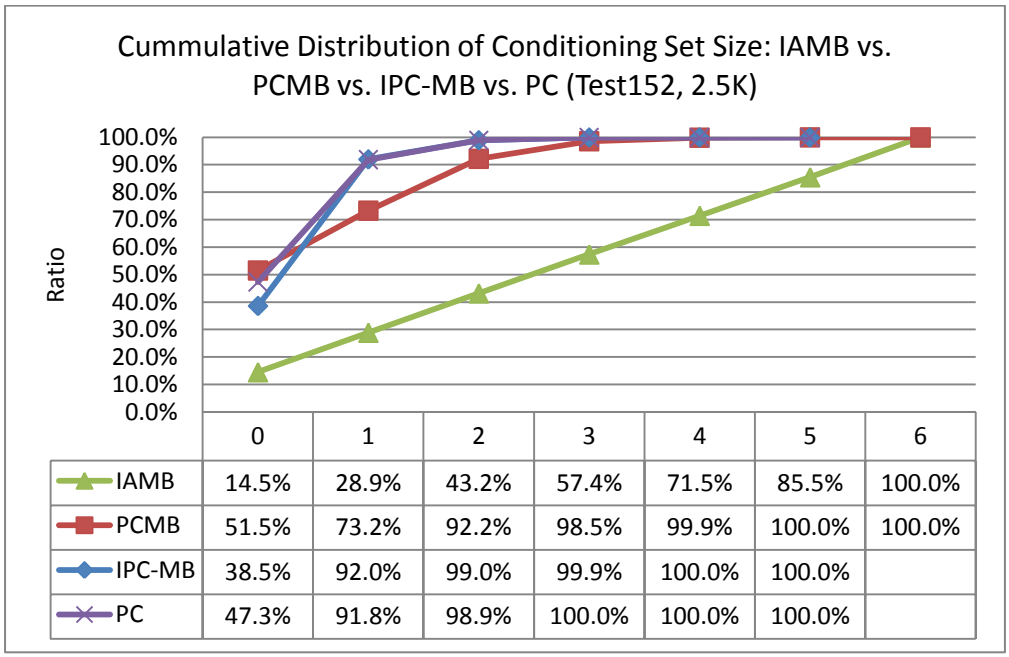
	0	1	2	3	4	5	6	7
IAMB	18.3%	36.1%	53.3%	69.9%	84.6%	95.9%	99.9%	100.0%
PCMB	12.3%	36.0%	70.6%	92.0%	99.1%	99.8%	100.0%	100.0%
IPC-MB	14.6%	82.7%	96.5%	99.9%	100.0%			
PC	24.6%	84.4%	97.3%	99.8%	100.0%			

Figure 4-20: Example distribution of conditioning set size (i.e. the cardinality of conditioning set) as involved in CI tests conducted by IAMB, PCMB, IPC-MB and PC in experiments of Alarm (The upper graph is the average distribution given 500 instances, and the bottom is that measured given 5,000 instances).



	0	1	2	3	4	5	6
PCMB	48.7%	72.5%	94.3%	98.9%	99.8%	100.0%	100.0%
IAMB	21.0%	41.4%	60.5%	78.6%	93.1%	99.8%	100.0%
IPC-MB	54.4%	93.2%	97.9%	99.6%	100.0%		
PC	59.6%	93.3%	98.1%	99.6%	100.0%		

Figure 4-21: Example distribution of conditioning set size (i.e. the cardinality of conditioning set) as involved in CI tests conducted by IAMB, PCMB, IPC-MB and PC in experiments of polytree version Alarm (5,000 instances).



	0	1	2	3	4	5	6
IAMB	14.5%	28.9%	43.2%	57.4%	71.5%	85.5%	100.0%
PCMB	51.5%	73.2%	92.2%	98.5%	99.9%	100.0%	100.0%
IPC-MB	38.5%	92.0%	99.0%	99.9%	100.0%	100.0%	
PC	47.3%	91.8%	98.9%	100.0%	100.0%	100.0%	

Figure 4-22: Example distribution of conditioning set size (i.e. the cardinality of conditioning set) as involved in CI tests conducted by IAMB, PCMB, IPC-MB and PC in experiments of Test152 (2,500 instances).

Figure 4-21 shows the distribution of conditioning set size as measured in experiments with the polytree version of Alarm, where 5,000 instances are available. Compared with the measures shown in Figure 4-20, it is noticed that the portion of CI tests with smaller conditioning set increases, for each algorithm covered. Meanwhile, many fewer number of CI tests are observed by comparing Table 4.8 and Table 4.11. Hence, we can conclude that problems of sparse networks are easier to solve.

4.7 Summary

A series of experiments with classical problems, ranging from small to large scale, are conducted, over IAMB, PCMB, IPC-MB and PC algorithms. By feeding different size of observations to these four algorithms, we study their relative performance in term of accuracy, time efficiency and data efficiency. Compared with IAMB, IPC-MB achieves much higher accuracy given the

same amount of samples, and the extra requirement on time is affordable; compared with PCMB, IPC-MB reaches the same or slightly higher accuracy but in much faster speed; compared with PC, IPC-MB demonstrates obvious advantage in term of time complexity as an algorithm requiring only local search.

In next chapter, we will go a little beyond the results collected in this chapter, discussing more on the causes behind the scene.

Chapitre 5 TRADEOFF ANALYSIS OF DIFFERENT MARKOV BLANKET LEARNING ALGORITHMS

5.1 Introduction

The necessary background of Markov blanket, existing learning algorithms, and our own proposed one are covered in previous chapters. A series of experiments are also designed to provide a vivid and direct comparison of their relative performance. In this chapter, we will go beyond the facts as presented in Chapter 4, with emphasis on why one algorithm behaves like what it appears, and we expect to end with a non-biased recommendation on one most appropriate algorithm for inducing Markov blanket.

5.2 Category of Algorithms

From 1996 on, there are at least 10 algorithms for inducing Markov blanket have been proposed, including KS, GS, IAMB and its variants (InterIAMB, InterIAMBnPC and Fast-IAMB), MMPC/MB, HITON-PC/MB, PCMB and our own IPC-MB. They actually can be classified into two groups:

1. Algorithms built on the property that $I(T, X|\mathbf{MB}_T)$, for $\forall X \in \mathbf{U}\backslash\mathbf{MB}_T$. KS, GS, IAMB and its variants belong to this category. We use ***GROUP I*** to refer to these algorithms in the remaining text of this chapter;
2. Algorithms built on the property that $\mathbf{MB}_T = \mathbf{PC}_T \cup \mathbf{Sp}_T$ and the underlying connection between $X \in \mathbf{MB}_T$ and the target T, i.e. the so-called topology information. More recent algorithms like MMPC/MB, HITON-PC/MB, PCMB and IPC-MB fall into this class. ***GROUP II*** is used to denote them in this chapter.

Algorithms of both categories depend on a series of conditional independence tests in the search of $\mathbf{MB}_T$.

In this chapter, we only consider IAMB, PCMB and IPC-MB considering that (1) IAMB and PCMB are representatives of ***GROUP I*** and ***GROUP II*** respectively; (2) Both are proved correct, and their relative performance data are collected in Chapter 4; and (3) All three algorithms require the same assumption, faithfulness.

5.3 Efficiency Gain by Local Search

Local search is defined relative to global search. Given the faithfulness assumption, if an algorithm could induce $\mathbf{MB}_T$ without having to induce the whole Bayeisan network over **U**, it is viewed as local search, or local learning. Based on this definition, IAMB, PCMB and IPC-MB all belong to this category.

However, it doesn't mean that local search is guaranteed to be more time efficient than global search. From the study in Section 4.5 (more specifically, Table 4.12), it is observed that IAMB and IPC-MB are able to achieve obvious reduction in time complexity as compared with PC (but IAMB performs much worse than PC in term of accuracy), and the gain is expected to be more remarkable with increasing scale of problems. Although PCMB produces as correct outcome as IPC-MB and PC given the same number of instances, its timing cost may even exceed that of PC (see Section 4.5.1, 4.5.2 and 4.5.3). Even though, we prefer to say the PC is such an excellent algorithm, instead of declaring that PCMB is not good enough.

5.4 Data Efficiency

5.4.1 Data Efficiency is Critical

As we see in Chapter 4, though one algorithm, like IAMB, can be correct theoretically, it may produce very poor results with limit instances. Normally the lower accuracy achieved by one algorithm given specific number of instances, the more data inefficient this algorithm is. One may argue for more observations to reach a satisfactory level, this is not realistic in real applications. For example, given the Alarm problem, even when 20,000 instances are allowed for IAMB, its accuracy is still much poorer than that reached by PCMB and IPC-MB given 5,000 instances. Therefore, IAMB has limit in applications, though it is expected to be time efficient.

Data efficiency is the most problem existing in ***GROUP I*** algorithms, and it is this problem which has attracted several following effort since the birth of IAMB.

5.4.2 Why IAMB is Very Data Inefficient

There are two reasons to cause the data inefficiency of IAMB. Firstly, IAMB and other algorithms in ***GROUP I*** depend on checking if $I(T, X|\mathbf{MB}_T^C)$ to determine whether or not add into

or remove from $\mathbf{MB}_T$ some variable X. This is direct and simple; however, they may condition on the whole $\mathbf{MB}_T$ or even larger one, so the number of instances required for reliable test then would be considerable. In fact, even if we have large samples, we still want the freedom degree of statistical tests be as small as possible to have more reliable tests.

Secondly, many false positives are added in the growing phase, which prevents true ones from being added. IAMB is composed of two phases: growing and shrinking. In the growing phase, variable X is added into $\mathbf{MB}_T^C$ if it is not conditionally independent of T given the current $\mathbf{MB}_T^C$. Sicne $\mathbf{MB}_T^C$ starts with empty on, obviously, false positives will be added, and they stay in $\mathbf{MB}_T^C$ since then. To make things worse, upon the first false one being allowed into $\mathbf{MB}_T^C$, the door is opened for more false positives. If we have too few instances, we may terminate the learning somewhere, ending with a possibly completely wrong $\mathbf{MB}_T^C$ candidate set, which was observed in our experiment. Feeding such a set into the shrinking phase can be a disaster, even worse than conditioning on the whole $\mathbf{MB}_T$. Therefore, pretty low precision and recall are observed on IAMB in our experimental studies.

5.4.3 PCMB is Data Efficient

The data inefficiency problem was noticed by others, including the authors of PCMB, so the growing and shrinking are interleaved in *GetPCD* of PCMB. In *GetPCD*, a best candidate is selected based on a series of conditional independence tests; upon one new candidate being added to *PCD*, all variables of *PCD*, including the one just added, are checked to see if there is any false positive. By doing so, false ones are recognized and removed in time, preventing error from being accumulated and resulting with more error as in IAMB.

Besides, by dividing the recognition of $\mathbf{MB}_T$ into $\mathbf{PC}_T$ and $\mathbf{Sp}_T$, the possibly largest conditioning set is further limited. Therefore, although PCMB works like IAMB by considering which ones should be included into $\mathbf{MB}_T$, it makes each decision with enough caution aiming at correctness as well as data efficiency.

5.4.4 IPC-MB is Data Efficient Too

Though IPC-MB has a similar framework as PCMB, it recognizes the $\mathbf{PC}_T^C$ in a quite different way. Instead of deciding which ones can be added into $\mathbf{PC}_T^C$ or $\mathbf{MB}_T^C$ as PCMB or IAMB does,

IPC-MB realizes this target by removing those known as false positives, with true ones left. This is built on the observation that false positives normally occupy a much larger portion among the whole attribute set **U**, e.g. the largest Markov blanket in Alarm is of size eight while totally there are 37 attributes. In IPC-MB, each false positive is found with a conditioning set of the smallest cardinality (**Theorem 3.13**). Similar to PCMB, the recognition of $\mathbf{MB}_T$ is divided into $\mathbf{PC}_T$ and $\mathbf{Sp}_T$ as well. By minimizing the number of variables in separator set in *FindCanPC*, the cardinality of the conditioning set involved in the recognition of a true spouse is also minimized, which further ensures the reliability of the algorithm (**Theorem 3.14**).

As shown in Section 4.6, PCMB and IPC-MB demonstrate obvious advantage over IAMB in term of data efficiency. Although the gain of accuracy of IPC-MB is not so obvious over PCMB in our experiments, the distributions of conditioning set cardinality as shown in Figure 4-20, Figure 4-21 and Figure 4-22 support such an argument: IPC-MB is expected to produce more reliable results than PCMB because it requires smaller conditioning set by average.

5.5 Time Efficiency

5.5.1 IAMB is Fast but with High Cost

The study in Chapter 4 shows that IAMB is the fastest one, and PCMB is terribly slow as compared to the other two. IPC-MB is slower than IAMB, but in an affordable scale.

In IAMB, the numbers of CI tests and data passes as required in both phases grow in a linear speed. Given **U**, both the number of CI tests and data passes needed in the worst case (Figure 3-5) are $(|\mathbf{U}| - 1)$ for the growing phase. Actually, it is also $(|\mathbf{U}| - 1)$ for the shrinking phase, and hence the total number is $2 * (|\mathbf{U}| - 1)$ for CI tests as well as data passes. Due to its data inefficiency, in practice, the actual number of CI tests and data passes may be even fewer than that as expected. The possible introduction of false candidates in the growing phase further make the thing worse as we discussed in 5.4.2. Therefore, in IAMB, CI tests quickly become un-reliable after few rounds in the growing phase, and the search has to terminate, with pretty poor results being produced.

5.5.2 IPC-MB is Much More Efficient Than PCMB

IPC-MB is observed to have slightly better accuracy performance than PCMB, but with much less cost in term of time. Both of them declare as local search, and they all include the topology information into consideration, then why they differ so great in time complexity? We analyze the cause from two aspects.

First, the authors of PCMB didn't realize the conclusion made on **Theorem 3.4**. Then, in PCMB, *GetPC* is called on not only the target T , but each $X \in \mathbf{PC}_T$. However, in IPC-MB, *FindCanPC*, which functions similarly as *GetPCD*, is called on T as well as each $X \in \mathbf{PC}_T^C$. All these *FindCanPC* calls combined together actually equals to one-time call of *GetPC*. Therefore, at least $|\mathbf{PC}_T|$ times of *GetPC* are saved in IPC-MB, which is significant considering that each *GetPC* is really time consuming.

Second, what search conducted in *FindCanPC* is more efficient than *GetPCD*. Given the example of Figure 3-5, the first step "remove false positives from *CanPCD*" (Figure 1-11) in PCMB has the same complexity as the whole *FindCanPC* in IPC-MB, that is $|\mathbf{U}| * 2^{|\mathbf{U}|-1}$; besides, each *GetPC* has the complexity as the recognition of $\mathbf{PC}_T$ in IPC-MB, that is $|\mathbf{U}|^2 2^{|\mathbf{U}|-1}$. Since PCMB calls *GetPC* for each $X \in \mathbf{PC}_T$, the corresponding complexity increases to $|\mathbf{U}|^3 2^{|\mathbf{U}|-1}$. If we count the extra two steps in *GetPCD*, i.e. "add the best candidate to *PCD*" and "remove false positives from *PCD*" (Figure 1-11), the whole complexity of PCMB will be even higher.

Therefore, PCMB loses to IPC-MB in time efficiency due to three causes:

1. *GetPCD* (PCMB) is much more complex than *FindCanPC* (IPC-MB). *GetPCD* actually follows the design of IAMB, but it interleaves the growing and shrinking to remove any false positives wrongly recognized at an early time. The adding of one possible candidate within each iteration is accompanied with two times of consuming checking, i.e. "remove false positives from *CanPCD*" and "remove false positives from *PCD*";
2. In *GetPCD*, the *PCD* not only differs between adjacent iteration, but within the same iteration. Based on the guideline of our implementation, i.e. only known contingency tables are constructed in one scanning of data file, we may need three data passes in each iteration of *GetPCD*, which further makes thing worse;

3. The needless calling of *GetPC* for each $X \in \mathbf{PC}_T$. *GetPC(X)* is called to collect parents and children of X in PCMB, and it has the same effect of what Line 2-12 do in IPC-MB. In other words, we only call *GetPC(X)* for one time in IPC-MB because we recognized that spouse candidates can be prepared meanwhile (**Lemma 3.5**), and only true spouses will pass the test $I_D(T, Y|Sepset \cup \{X\})$ (**Theorem 3.4**). Therefore, those *GetPC(X)* for $X \in \mathbf{PC}_T$ is not necessary, and it greatly increases the whole complexity of PCMB considering that each individual *GetPC(X)* is so time consuming a procedure.

Therefore, to achieve better performance over IAMB and other previous works, IPC-MB is paid with more affordable additional cost than PCMB.

5.6 Scalability

If we view the data as a matrix, with columns for features and rows for instances, scalability refers to the ability that one algorithm works well given high dimensionality (column-wise), or large number of observations (row-wise), or both.

Regarding IAMB, it doesn't consider the topology, so the number of dimensions, or features, directly influences its actual performance. In our experiments, acceptable accuracy level is only observed in Asia problem. Though we believe that given enough instances, IAMB is able to produce perfect results, the number of instances as required may be too large to meet. Therefore, IAMB is not expected to perform well given increasing dimensionality, except when there are also considerable observations accordingly.

PCMB is shown indeed more data efficient than IAMB, i.e. producing much higher accuracy given the same amount of observations. Given fixed dimensionality, PCMB is also expected to achieve much faster increase in accuracy than IAMB. However, as one algorithm declaring local search, PCMB is quite time inefficient, and it may even cost much more than the global search by PC algorithm. Though PCMB is shown scalable by its author in [13], where it is applied to a problem with 139,351 features appearing in KDD-Cup'2001, we doubt the conclusion very much.

IPC-MB inherits the advantage of PCMB, i.e. data efficient. In fact, it may be more data efficient than PCMB, as shown in our experiments (refer to Section 4.6.2). However, IPC-MB runs in a much faster speed than PCMB. In Table 5.1, we observe that IPC-MB achieve the same or better

result than PCMB, requiring many fewer CI tests. Compared with IAMB, the additional cost for IPC-MB is affordable if we realize the much higher accuracy as achieved by IPC-MB.

Table 5.1: The comparison of IPC-MB to PCMB and IAMB in terms of time efficiency and accuracy. About time cost, ↑ x% means IPC-MB costs x% more CI tests than PCMB or IAMB; and about accuracy, ↑ x% means IPC-MB's distance to the perfect result is x% larger than PCMB or IAMB (note: the smaller the distance, the more accurate the result).

Problem (# instances)	**PCMB**		**IAMB**	
	Time Cost (CI Tests)	**Accuracy (Distance)**	**Time Cost (CI Tests)**	**Accuracy (Distance)**
Asia (20K)	↓ 83.0%	↓ 16.7%	↑ 296.0%	↓ 33.3%
PolyAlarm (5K)	↓ 95.1%	↓ 0.0%	↑ 14.6%	↓ 74.4%
Alarm (5K)	↓ 95.1%	↓ 16.7%	↑ 344.7%	↓ 92.9%
Test152 (2.5K)	↓ 76.2%	↓ 36.4%	↑ 47.0%	↓ 87.5%

Both PCMB and IPC-MB are sensitive to the underlying topology, or we can say that the actual topology influences their scalability a lot. For example, although both Alarm and PolyAlarm have 37 attributes, the actual timing cost by IPC-MB and PCMB differs greatly. In addition, we observe that IPC-MB and PCMB achieve quite expressive results in Test152 problem given only 2,500 instances; they need 5,000 or more instances to reach the same accuracy level in Alarm problem, though Test152 has much higher dimensionality than Alarm problem.

In conclusion, compared with IAMB and PCMB, IPC-MB achieve a good tradeoff by improving the data efficiency with reasonable additional timing cost; hence, it is expected to have better scalability. Besides, the underlying topology influences the actual scalability of algorithms depending on the structure, like PCMB and IPC-MB.

5.7 Information Deduced

To the problem of feature selection, recognizing the variables belonging to the Markov blanket of T is the target. Regarding this goal, all three algorithms under study are known with this ability from the theoretical viewpoint, i.e. producing the perfect result given enough information.

However, given the faithfulness assumption, Markov blanket is known as unique and it contains target's parents, children and spouses (along with edges and corresponding orientations). IAMB and other previous algorithms only recognize that variables of MB render the rest of variables independent of target; their designs are built on this property, and their output only tell us if

$X \in \mathbf{MB}_T$ given $X \in \mathbf{U}$. PCMB and IPC-MB are built on the information as encoded in the underlying connectivity existing on Markov blanket and T. It not only enables algorithms to achieve much better data efficiency than IAMB but generate more informative result. IAMB doesn't distinguish the parents, children and spouses of T, but PCMB and IPC-MB separate spouses from the remaining variables in $\mathbf{MB}_T$. Besides, those common children of spouses and T are recognized among $\mathbf{MB}_T \backslash \mathbf{Sp}_T$, so as the orientations of related arcs, see Figure 5-1.

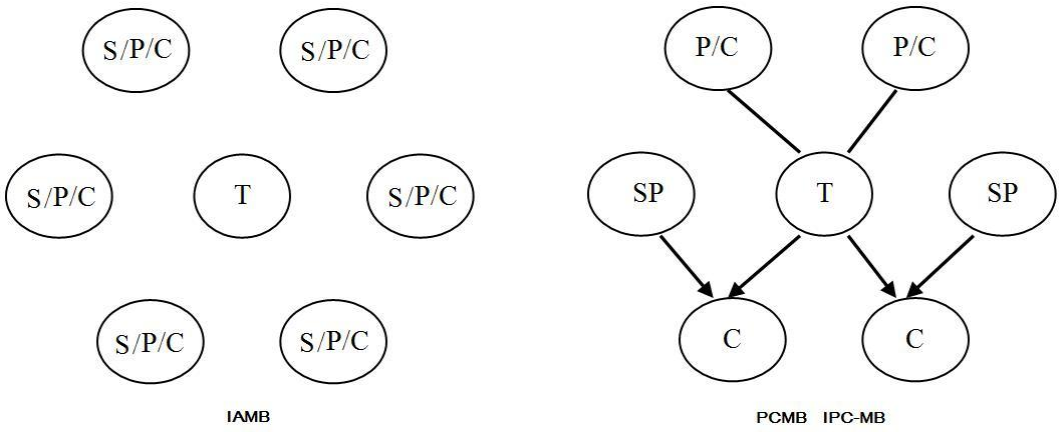


Figure 5-1: Output of IAMB (left), PCMB and IPC-MB (right)

The additional information as found by PCMB and IPC-MB may be helpful for applicants to understand the underlying problem better. Furthermore, they can be made use to reduce the effort for the learning of Markov blanket classifier as to be discussed in Chapter 6 and 7.

5.8 Approximate Version of IPC-MB

Although IPC-MB is demonstrated very efficient in our experiments, its time complexity may still be un-affordable given network with dense connections or with large Markov blanket. For example, as we discuss in Section 3.9, it has to continue the search until there are no conditional independence tests left undone, even all false positives are able to be removed with conditioning sets of size smaller than two, given polytree networks. By imposing the checking of reliability of tests, as found in Figure 4-5 (Section 4.3.2), the search may terminate at earlier time when there are no more trustable tests available, which possibly reduces the time complexity though it is added originally to guarantee the correctness of results in practice.

However, we still may face large search space when there are ample instances for learning. One common choice as referred in conditional test based structure learning algorithms of Bayesian network can be applied here to reduce the search space to an expected level, i.e. restricting the maximum number of parents. In IPC-MB, this is equal to limit the maximum value of *cutSetSize* in *FindCanPC* because IPC-MB, actually, depends the Markov condition to remove those non-descendants in *FindCanPC*. Therefore, we have a new version of *FindCanPC* (Figure 5-2) derived from the version proposed in Figure 4-5.

```
FindCanPC(T: Target, PC_T^C: Candidate PC, D: Dataset, ε: SignificanceValue)
{
1.  NonPC = ∅;
2.  cutSetSize = 0;
3.  repeat
4.    notReliableAnyMore = true;
5.    for( ∀X ∈ PC_T^C) do
6.      for(S ⊆ PC_T^C\{X}) with |S| = cutSetSize) do
7.        if(I_D(T, X|S) is reliable) then
8.          notReliableAnyMore = false;
9.          if(I_D(T, X|S) ≤ ε) then
10.           NonPC = NonPC ∪ {X};
11.           Sepset_{T,X} = S;        //Cache for later reference
12.           break;
13.         end if
14.       end if
15.     end for
16.   end for
17.   PC_T^C = PC_T^C \ NonPC;
18.   NonPC = ∅;
19.   cutSetSize ++;
20. until(cutSetSize>maxNumParents or |PC_T^C| ≤ cutSetSize or notReliableAnyMore = true)
21. return PC_T^C;
}
```

Figure 5-2: The version of FindCanPC that restricts the search space as well as considers reliability of statistical tests.

Though it is an approximate version, it doesn't mean that we won't get correct results. For example, with a polytree, if we set the maximum number of parents as two or larger value, the

outcome of *FindCanPC* as well as IPC-MB are both guaranteed. Considering that real problems generally have spare connectivity, most false positives can be recognized and removed given small conditioning set. Therefore, this approximate version will not entail a large loss of accuracy, but both the time and space complexity are reduced to an predictable level.

PCMB may have an approximate version by applying the same limit on the number of parents. However, there is no such choice for IAMB since it is not dependent on the underlying topology. The only possible choice is to limit the maximum cardinality of target $\mathbf{MB}_T$, which will make the performance of IAMB worse.

5.9 Summary

Although IAMB, PCMB and IPC-MB are all proved correct theoretically, they still demonstrate relative strength or weakness when applied to real problems, as revealed by the experiments conducted in Chapitre 4. In this chapter, we go beyond the facts observed, aiming at deciphering some causes existing behind the facts.

For practical applicants, based on our experience, IAMB is strongly recommended if there are ample data because it is easy to implement, fast in speed and efficient in memory usage. However, we should realize that the need for large data samples increases quickly (actually, exponentially) when the number of variables and/or the number of values per variable increase, and rarely we can meet the corresponding requirement. In our experiments with IAMB, satisfactory result is observed only on Asia, one very tiny problem containing only eight variables.

Compared with IAMB, PCMB and IPC-MB have much better accuracy performance given the same amount of observations, which reflects their data efficiency property. However, PCMB is much slower than IPC-MB, and it may cost more time than PC. Hence, IPC-MB is further recommended over PCMB. Considering that samples are always small relative to the observation space, i.e. $\prod X_i$ where $X_i \in \mathbf{U}$, and the time cost is affordable, IPC-MB is determined as the best choice among the three for applicants. Approximate version is possible for PCMB and IPC-MB, but not IAMB. Table 5.2 gives a brief summary on their relative features.

Table 5.2: Trade-off summary over IAMB, PCMB and IPC-MB.

	IAMB	**PCMB**	**IPC-MB**
Assumption(s)	Faithfulness	Faithfulness	Faithfulness
Local Search	Yes, and less cost than PC	Yes, but cost may be higher than PC	Yes, and less cost than PC
Data Efficiency	Very poor	Good	Best
Time Efficiency	Best	Poor	Good
Scalability	Ignored	Applicable(especially with approximation version)	Applicable(especially with approximate version)
Information Induced	Only $\mathbf{MB}_T$	$\mathbf{MB}_T$ plus partial connections and orientations	$\mathbf{MB}_T$ plus partial connections and orientations
Implementation Difficulty	Simple	Difficult, and should pay attention on the code and memory optimization	Simple but should pay attention to memory usage optimization

Chapitre 6 A NOVEL LOCAL LEARNING ALGORITHM OF BAYESIAN NETWORK CLASSIFIER: IPC-MBC

We have shown how to derive the set of variables that compose a Markov Blanket with local learning algorithms in the previous chapters. This set of variables can be used for any classifier and it is known as the optimal set for that purpose. However, the full topology of the BN over tihs set of nodes is not derived from these algorithms. This topologhy corresponds to a Markov Blanket Classifier. In this chapter, we introduce a novel algorithm for efficiently learning this topology. Compared with conventional structure learning algorithms, e.g. PC, we limit the search in a local manner as we do in IPC-MB, so obvious reduction of time cost is observed.

6.1 Background

Classification is a fundamental task in data mining that requires learning a classifier through the observation of data. Basically, a classifier is a function that maps instances described by a set of attributes to a class label. Naïve Bayes networks have been widely used for the task of classification [43, 44] (Figure 6-1 upper-left). They represent a special case of the more general Bayesian networks (BN) formalism and are characterized by their strong assumption about the independence of attributes given the target node. Although they generally perform fairly well in spite of this assumption [6], they lack the power to represent more complex dependencies among attributes and the target node that can affect performance. Tree Augmented Naïve Bayes [7] (Figure 6-1 upper-right) is an extension of Naïve Bayes that weakens its assumption, allowing additional dependence relations among attributes. It is empirically shown to yield better performance [7].

Compared with Naïve Bayes and TAN, a BN (Figure 6-1 bottom) doesn't distinguish between target (class) variable and attributes. The target can be a parent or child of attributes, and general dependencies are found among attributes. Although such general BN is expected with several promising merits, including (1) yielding better performance than Naïve Bayes and TAN [8], (2) encoding more detailed dependence relations as needed in diagnosis applications, and (3) inferring any node's possible state given complete or incomplete observations of other nodes, the NP-complete complexity to learn a BN inhibits its widespread application.

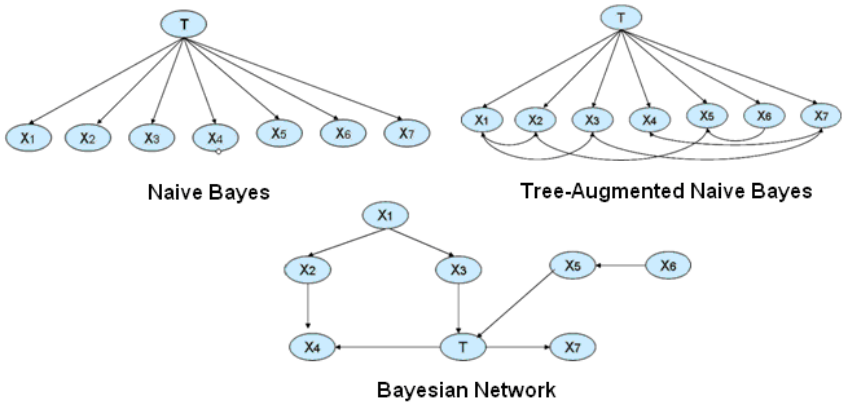


Figure 6-1: Examples of Bayesian classifiers, including Naïve Bayes (upper left), Tree-Augmented Naïve Bayes (upper right) and Bayesian Network (bottom)

However, we note that not all attributes are effective in predicting the target in applying BN as a classifier. With the BN example in Figure 6-1 (bottom), we have a decision rule like,

$$P(T|X_1,\dots,X_7) \propto P(T,X_1,\dots,X_7)$$
$$= P(T|X_3,X_5)P(X_3|X_1)P(X_5|X_6)P(X_2|X_1)P(X_4|X_2,T)P(X_7|T)P(X_1)P(X_6)$$

(6.1)

of which some terms, namely $P(X_3|X_1), P(X_5|X_6), P(X_2|X_1), P(X_1), P(X_6)$ do not contain the target variable T, which means that their values have no direct influence on the classification decision of T. By removing them, we obtain a simpler decision rule with no sacrifice with regards to classification performance:

$$\underset{t}{\operatorname{argmax}}\, P(T=t|X_3,X_5)P(X_4|X_2,T)P(X_7|T)$$

(6.2)

The attributes $\{X_2, X_3, X_4, X_5, X_7\}$, involved in this new version of the decision rule (6.2) correspond to the Markov blanket of T, i.e. $\mathbf{MB}_T$. Actually T, $\mathbf{MB}_T$ plus the arcs among them also constructs a Bayesian network, part of the original whole BN, and it encodes all the dependence relationships appearing in (6.2). Obviously, if we have the Bayesian network over **U**, it is trivial to get the sub-network that is effective for the classification of T, and it is called Markov blanket classifier (MBC) (with another name Bayesian network classifier (BNC) in one of our early publication [15]) this article to distinguish it from the whole BN over **U**.

Definition 6.1 (Markov Blanket Classifier or Bayesian Network Classifier). Given a Bayesian network over **U**, the partial DAG over $T \cup \mathbf{MB}_T$ is called the Markov Blanket Classifier, or Bayesian Network Classifier about T, and denoted as $\mathbb{MBC}_T$ or $\mathbb{BNC}_T$.

Note : (1) We will use MBC or BNC to refer the general concept; (2) Because BNC was used in our early published work [15], it is mentioned for easy reference, though MBC is preferred considering its close connection with Markov Blanket.

As mentioned above, with a learned (known) BN and a given target T, getting the target $\mathbb{MBC}_T$ is a trivial task. In this chapter, we propose one algorithm to learn the $\mathbb{MBC}_T$ without having to learn the whole BN first. It is built on IPC-MB, and it is proved correct, demonstrated as more efficient than the conventional approaches which have to learn the BN before we can get MBC. In Section 6.2, necessary knowledge of Bayesian network is covered for later reference and self-contained purpose. Then, in Section 6.3, how the related work is motivated is introduced in brief. In Sections 6.4, a local structure learning algorithm for MBC is proposed, and its correctness is proved. The complexity analysis is conducted in Section 6.5, followed by empirical study and discussion in Section 6.6 and 6.7. A brief conclusion about this chapter is made with Section 6.8.

6.2 Structure Learning of Bayesian Network

Since MBC is a BN, but over a feature subset $\mathbf{MB}_T \sqsubseteq \mathbf{U}\backslash\{T\}$, those methods applicable to the structure learning of BN are believed useful references for our work. There are two ways to view a Bayesian network, each suggesting a particular approach to learning and they are described below.

6.2.1 Conditional Independence Test Approach

This approach views the BN as a structure that represents a group of conditional independence relationships among the nodes, according to the concept of d-separation [2]. This suggests learning the BN structure by identifying the conditional independence relationships among the nodes. Using some statistical test (such as Chi-squared test), we can find the conditional independence relationships among the attributes and use these relationships as constraints to construct a BN. These algorithms are referred as CI-based algorithms or constraint-based algorithms [14, 23, 35, 36]. This approach includes IC algorithm (inductive causation) [2], PC

algorithm (after its authors, Peter and Clark) [14], GS algorithm (grow and shrink) [23] and TPDA algorithm (three-phase dependency analysis) [36]. All of them recover structures to be consistent with the conditional independencies among the variables. Generally, algorithms start by learning the skeleton of the graph (by propagating constraints on the neighborhood of each variable) and then edges are oriented to cope with dependencies revealed from data. Finally, one network is retained from the equivalent class consistent with the series of tests. Under the faithfulness condition, such strategies have been proven to build a graph converging to the true network as the size of the data approaches infinity. Moreover, their complexity is polynomial, assuming that the maximal degree of the network, that is, the maximal size of direct neighbors, is bounded [45].

6.2.2 Score-and-Search Approach

The second approach views the BN as a structure that represents the joint distribution of the attributes. This suggests that the best BN is the one that best fits the data, and leads to the scoring-based learning algorithms, that seek a structure maximizes the Bayesian, MDL or Kullback-Leibler (KL) entropy scoring function [46, 47]. Since the search space is known to be of a super exponential size on the number of nodes n, that is, $O\left(n!\,2^{\binom{n}{2}}\right)$ [48], an exhaustive search is practically infeasible, implying that various greedy strategies have been proposed to browse DAG space, sometimes requiring some prior knowledge.

Among them, the state-of-the-art greedy hill climbing strategy, although simple and yielding only a locally optimal network, remains one of the most employed method in practice, especially with larger networks. There exist various implementations using different empirical tricks to improve the score of the results, such as Tabu List, restarting, simulated annealing or searching with different orderings of the variables [49, 50].

No matter what scoring function to take and what heuristic to employ during the search, such algorithm will process in the following manner:

1. Start the search from a given DAG, usually the empty one or Naïve Bayes network;

2. Then, from a list of possible transformations containing at least addition, withdrawal or reversal of an edge, select and apply the transformation that improves the score most while also ensuring that graph remains acyclic;
3. Finally repeat previous step until strict improvements to the score can no longer be found.

6.2.3 Statistical Equivalence

A Bayesian network structure $\mathbb{G}$ represents conditional independence assumptions that allow the joint distribution to be decomposed, reducing the number of parameters. The graph $\mathbb{G}$ encodes the Markov assumption: Each variable X_i is independent of its non-descendants, given its parents in $\mathbb{G}$.By applying the chain rule of probabilities and properties of conditional independencies, any joint distribution that satisfies the Markov assumption can be decomposed into the product form

$$P(X_1, \dots, X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i)) \tag{6.3}$$

The Bayesian network structure $\mathbb{G}$ implies a set of independence assumptions in addition to (6.3). Let $I(\mathbb{G})$ be the set of independence statements (of the form X is independent of Y given Z) that hold in all distributions satisfying these Markov assumptions, and they can be derived as consequences of (6.3) [2].

More than one graph can imply exactly the same set of independencies. For example, consider a BN over two variables X and Y. The graphs $X \rightarrow Y$ and $X \leftarrow Y$ both imply the same set of independencies (i.e., $I(\mathbb{G}) = \emptyset$). Two graphs G and G' are equivalent if $I(\mathbb{G}) = I(\mathbb{G}')$[51]. That is, both graphs are alternative ways of describing the same set of independencies.

This notation of equivalence is crucial since when we examine observations from a distribution, we cannot distinguish between equivalent graphs. Pearl and Verma [29] show that we can characterize equivalent classes of graphs using a simple representation. In particular, these results establish that equivalent BNs have the same underlying undirected graph but might disagree on the direction of some of the arcs.

Theorem 6.1 Two DAGs are equivalent if and only if they have the same underlying undirected graph and the same v-structures (i.e. same set of uncoupled head-to-head converging, such as $X \rightarrow Y \leftarrow Z$) [29, 52].

This theorem implies that (1) learning the v-structures is critical for learning a Bayesian network, and (2) the remaining arcs' directions have no influence on the usage.

6.3 Motivation, Heuristics and Our Work

With a known BN over **U** and T of interest, it is trivial to extract the target $\mathbb{MBC}_T$. However, from the discussion in Section 6.2 and our experimental study in Section 4.5 (on PC), it is known that both CI and score-and-search approaches fail to scale to large problems. Actually, compared with the whole network, $\mathbb{MBC}_T$ normally occupies quite a small area of the whole DAG, in terms of both nodes and arcs. For example, the largest MBC in Asia, Alarm, Hailfinder, Test152 and PolyAlarm has 5, 8, 8, 17 and 5 nodes respectively, as compared with 8, 37, 56, 152 and 37 nodes as contained in the corresponding network. Therefore, an ideal solution permitting to learn only the nodes and arcs related with the target $\mathbb{MBC}_T$ is expected. Having to learn the DAG over **U** first is not what we prefer, though we have no other choice in the past, since it means waste of computing resource; and generally, the larger is the whole DAG, the more is the possible waste. How to reduce search space and reach an efficient learning algorithm for MBC is the goals of this and next chapter.

Given the output of IPC-MB, we want to make use of it to solve the pending problem considering three facts that (1) IPC-MB enables us to find the correct $\mathbf{MB}_T$, (2) $\mathbf{MB}_T$ contains all and only the nodes belonging to the target $\mathbb{MBC}_T$ (except for T), and (3) $\mathbf{MB}_T$ is much smaller than **U**, restricting the remaining search in a quite smaller scope. With $\mathbf{MB}_T$ ready, all existing methods available for the structure learning of BN are applicable without changes. One typical naïve procedure is to apply IPC-MB first to recognize $\mathbf{MB}_T$, and then apply constrained (CI-test) or score-and-search learning algorithms as though $\mathbf{U} = \mathbf{MB}_T$. However, we can take advantage of some of the structure learning that occure in IPC-MB to derive $\mathbb{MBC}_T$ more efficiently.

6.4 IPC-MBC Algorithm Specification and Proof

6.4.1 Overall Description

IPC-MBC requires faithfulness assumption, and it also depends on a series of conditional tests to determine if any link $X - Y$ should exist or not. The overall design of IPC-MBC is based on such an fact – if we know $\mathbf{PC}_X$ of each $X \in \mathbf{MB}_T$, the union of links between X and $Y \in \mathbf{PC}_X \cap \mathbf{MB}_T$ should belong to the target $\mathbb{MBC}_T$.

Given target $T \in \mathbf{U}$ and observations D, the whole procedure of IPC-MBC (Figure 6-2) can be divided into five sequential steps as described below:

1. **Induce the connections between T and $\mathbf{PC}_T^C$** (Line 1-5). IPC-MBC starts with an initial G in which T is connected with $\forall X \in \mathbf{U}$. The false parents/children are removed by disconnecting them from T via the call of *FindCanPC-MBC*(T), with possible exception on T's descendants. Candidate parents/children of T then are retrieved from G based on the linkage, denoted as $\mathbf{PC}_T^C$. T is marked as scanned by adding it into the container $Scanned$;

2. **Remove false positives from $\mathbf{PC}_T^C$ to get $\mathbf{PC}_T$, add links between $\forall X, Y \in \mathbf{PC}_T$ and collect spouse candidates** (Line 6-10). Given $\forall X \in \mathbf{PC}_T^C$, it is initialized to be connected to all $Y \in \mathbf{U} \backslash Scanned$ in G (Note: the edges existing between X and $Y \in Scanned$ therefore are kept un-changed). Then *FindCanPC-MBC(X)* is called to remove false positives from its adjacent neighbors to get $\mathbf{PC}_X^C$. The current X is added to $Scanned$ as well. After such call of *FindCanPC-MBC(X)* on $\forall X \in \mathbf{PC}_T^C$, (1) what connected to T are only its true parents and children, denoted as $\mathbf{PC}_T$; (2) the edges existing between any pair of $X, Y \in \mathbf{PC}_T$ are added in G; and (3) nodes adjacent to $X \in \mathbf{PC}_T$ are known as candidate spouses, denoted as $\mathbf{SP}_T^C$;

3. **Recognize true spouses, $\mathbf{Sp}_T$, add links among $\mathbf{Sp}_T$, and between $\mathbf{Sp}_T$ and $\mathbf{PC}_T$** (Line $11 - 22$). We retrieve $\mathbf{PC}_T$ and $\mathbf{SP}_T^C$ first from G based on the connection. For $\forall X \in \mathbf{PC}_T$, we similarly retrieve $\mathbf{PC}_X^C$ where $\mathbf{PC}_X^C = \{Y | (X - Y) \in G\}$. Then, for $\forall Y \in \mathbf{PC}_X^C \backslash Scanned$, if it is dependent with T as conditioned on $Sepset_{T,Y} \cup \{X\}$, it is known as a true spouse. For such Y, we add links between it and each $Z \in \mathbf{U} \backslash Scanned$, and call *FindCanPC-MBC(Y)* to induce the links as may exist between Y and $Z \in \mathbf{Sp}_T \cup \mathbf{PC}_T$.

Since each true spouse Y is processed in the same way, we won't miss any links among $\mathbf{Sp}_T$, as well as links between $\mathbf{Sp}_T$ and $\mathbf{PC}_T$;

4. **Remove nodes not belonging to $\mathbf{PC}_T$ and $\mathbf{Sp}_T$** (Line 23). The arcs connecting to the removed nodes are deleted as well, with the skeleton of $\mathbb{MBC}_T$ and some know V-structures left in G;

5. **Orienting the arcs**. A series of orientation rules are applied to the outcome of Step 4 to get the final $\mathbb{MBC}_T$.

These five steps summarize the overall design of IPC-MBC (Figure 6-2), from which one can see that we repeatedly depend on the recognition of parents and children, via calling *FindCanPC-MBC*(Figure 6-3), to determine the connection between any pair of nodes (including Step 1, 2 and 3). This is similar to what we done in IPC-MB, but more complex since here we care not only $\mathbf{MB}_T$, but also about links existing among $\mathbf{MB}_T \cup \{T\}$. Because we carefully restrict the call of *FindCanPC-MBC* within a local scope by (1) following breadth-first manner, (2) removing confirmed false positives, and (3) preventing duplicate study, a great reduction of complexity is expected. In the coming sections, each step will be expanded with more details, along with necessary proof of correctness.

IPC − MBC(T: Target, D: Dataset, ε: Significance Value)
{
// Step 1: Recognize $CanPC(T)$
1. $\boldsymbol{Scanned} = \{\}$;
2. $G = \{(T - X) | \forall X \in \mathbf{U} \backslash \{T\}\}$;
3. $G = FindCanPC - MBC(T, G, D, \varepsilon)$;
4. $\mathbf{PC}_T^G = \{X | (T - X) \in G\}$;
5. $\boldsymbol{Scanned} = \boldsymbol{Scanned} \cup \{T\}$;
//Step 2: Recognize $\mathbf{PC}_T$/Add links among $\mathbf{PC}_T$
6. **for**($\forall X \in \mathbf{PC}_T^G$) **do**
7. $G = G \cup \{(X - Y) | \forall Y \in \mathbf{U} \backslash \boldsymbol{Scanned}\}$;
8. $G = RecognizePC - MBC(X, G, D, \varepsilon)$;
9. $\boldsymbol{Scanned} = \boldsymbol{Scanned} \cup \{X\}$;
10. **end for**
//Step 3: Recognize $\mathbf{Sp}_T$/Add links among $\mathbf{Sp}_T$/Add links between $\mathbf{Sp}_T$ and $\mathbf{PC}_T$
11. $\mathbf{PC}_T = \{X | (T - X) \in G\}$;
12. **for**($\forall X \in \mathbf{PC}_T$) **do**
13. $\mathbf{PC}_X^G = \{Y | (X - Y) \in G\}$;
14. **for**($\forall Y \in \mathbf{PC}_X^G$ *and* $Y \notin \boldsymbol{Scanned}$)**do**
15. **if**$\left(I_D(T, Y | Sepset_{T,Y} \cup \{X\}) > \varepsilon\right)$ **then**
16. Set $T \rightarrow X \leftarrow Y$ in G;
17. $G = G \cup \{(Y - Z) | \forall Z \in \mathbf{U} \backslash \boldsymbol{Scanned}\}$;
18. $G = RecognizePC - MBC(Y, G, D, \varepsilon)$;
19. $\boldsymbol{Scanned} = \boldsymbol{Scanned} \cup \{Y\}$
20. **end if**
21. **end for**
22. **end for**
//Step 4: Clean unnecessary nodes and links in G, with the skeleton of $\mathbf{MBC}_T$ left
23. Remove all X from G such that $X \notin (\mathbf{PC}_T \cup \mathbf{Sp}_T \cup \{T\})$ so as those links related to the removed nodes, with $\mathbf{MBC}_T$ skeleton plus some known v-structures left;
// Step 5: Orientation (refer to Section-**Orientation**)
24. **return** G;
}

Figure 6-2: The overall algorithm specification of IPC-MBC

```
FindCanPC − MBC(T: Target, G: Graph to work with, D: Dataset, ε: Significance Value)
{
1.  NonPC = {};
2.  cutSetSize = 0;
3.  PC_T^G = {X|(T − X) ∈ G};   //Retrieve adjacent nodes of T
4.  do
5.    for(∀X ∈ PC_T^G) do
6.      for(∀S ⊆ PC_T^G\{X} with|S| = cutSetSize) do
7.        if(I_D(T, X|S) ≤ ε)then
8.            G = G\{(T − X)};
9.            NonPC = NonPC ∪ {X};
10.           Sepset_{T,X} = S;      // Cache for reference in IPC-MBC
11.           break;
12.       end if
13.     end for
14.   end for
15.   PC_T^G = PC_T^G\NonPC;
16.   cutSetSize = cutSetSize + 1;
17. while(|PC_T^G| > cutSetSize)
18. return G;
}
```

Figure 6-3: *FindCanPC-MBC* algorithm specification.

6.4.2 Induce Candidate Parents/Children of Target

As the name of this algorithm indicates, the whole learning depends on the discovery of parents and children enables us to induce the links of interest, which is critical to the locality nature of this algorithm. *FindCanPC-MBC* procedure (Figure 6-3) is responsible for the learning of parent/child **candidates**, and it has four input parameters:

1. T, the active target that we are going to study its connectivity status with others;
2. G, the graph container which contains (1) what we have found, and (2) manually added adjacent neighbors of $T, i.e. \mathbf{U}\backslash\boldsymbol{Scanned}$, upton entering *FindCanPC-MBC*.
3. D, the dataset prepared for training;

4. ε, threshold value to be used in determining if a conditional independence test indicates "true" CI relationship or not, e.g. significant or not. It is set empirically, and common choice may be 0.01 or 0.05.

The output of *FindCanPC-MBC* is contained in G, but with some links possibly deleted compared with the state when it just enters into the function. **For easy reference purpose, we use** $G_i(i = 1..5)$ **to represent the graph as got by the end of each of the five step**.

FindCanPC-MBC(T, G) (the remaining two parameters are ignored since they are same for different calls) actually is same as *FindCanPC*$(T, \mathbf{PC}_T^C)$ in IPC-MB since $\mathbf{PC}_T^C$ can be retrieved from G, and it is actually done at Line 3 in *FindCanPC-MBC*. Although there is more information contained in G (except for the first call of *FindCanPC-MBC*), they, in fact, are ignored within *FindCanPC-MBC*. Therefore, all discussions and conclusions on *FindCanPC* (in 3.4. 1) apply here. In the first step, G is initialized as $\{(T - X)|\forall X \in \mathbf{U}\backslash\{T\}\}$, and we have the following two corollary derived from previous conclusions made in Chapitre 3.

Corollary 6.1 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, given $G = \{(T - X)|\forall X \in \mathbf{U}\backslash\{T\}\}$, *FindCanPC-MBC* enables us to find the superset of $\mathbf{PC}_T$, denoted as $\mathbf{PC}_T^C$ (Candidate Parents and Children), and $\mathbf{PC}_T \subseteq \mathbf{PC}_T^C$..

Proof. *Please refer to* **Theorem 3.1**. ▉

Corollary 6.2 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, given $G = \{(T - X)|\forall X \in \mathbf{U}\backslash\{T\}\}$, the only possible false positives as output by *FindCanPC-MBC* are T's descendants.

Proof. *Please refer to* **Theorem 3.2**. ▉

Therefore, by the end of the first step, we have $\mathbb{G}_1 = \{(T - X)|X \in \mathbf{PC}_T^C\}$, and Figure 6-4 gives an example illustration.

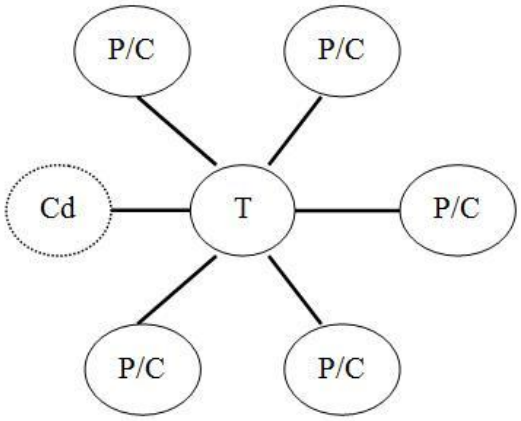


Figure 6-4: $\mathbb{G}_1$ contains all the parents and children of T (denoted as $\mathrm{P/C}$ since they cannot be distinguished for now) as connected to T, as well as some false positives possibly, i.e. children's descendants ($\mathrm{C_d}$ with dotted circle). Note that nodes NOT connected to T are not drawn in this graph.

6.4.3 Recognize $\mathbf{PC}_T$/Links among $\mathbf{PC}_T$/ $\mathbf{PC}_X^C$

Given $\mathbb{G}_1$, it is trivial to retrieve nodes direcly connecting with T in $\mathbb{G}_1$, denoted as $\mathbf{PC}_T^C$. In this step, we call *FindCanPC-MBC* on each $X \in \mathbf{PC}_T^C$. If we denote the graph to have by the end of this step as $\mathbb{G}_2$, we need to prove the following two findings:

- $\{X|(T-X) \in \mathbb{G}_2\}$ is exactly parents and children of T, i.e. $\mathbf{PC}_T$; Besides,
- $\{(X-Y)|(X-Y) \in \mathbb{G}_2\ , \forall\, X,Y \in \mathbf{PC}_T \ and\ X \neq Y\} = \{(X-Y)|(X-Y) \in \mathbb{MBC}_T, \forall\, X,Y \in \mathbf{PC}_T \ and\ X \neq Y\}$, i.e. the links existing among $\mathbf{PC}_T$ in $\mathbb{G}_2$ are exactly those among $\mathbf{PC}_T$ in the target $\mathbb{MBC}_T$.

Lemma 6.1 Given $G = G \cup \{(Y-X)|\forall X \in \mathbf{U}\backslash\boldsymbol{Scanned}\}$**,** the call of *FindCanPC-MBC*(Y, G) will output all parents and children of Y.

Proof. *Scanned is known to contain nodes having FindCanPC-MBC called. Then, given* $G = G \cup \{(Y-X)|\forall X \in \mathbf{U}\backslash\boldsymbol{Scanned}\}$, *(1)nodes contained in Scanned and confirmed not connecting to* Y *in* $\mathbb{MBC}_T$ *are excluded from consideration, as expected; and (2) nodes known as connecting to* Y *and contained in Scanned keep remained in* G*. Upon the calling of FindCanPC-MBC*(Y, G)*, all parents and children must connect to* Y *in* G *; otherwise, it means that some* $X \in \mathbf{PC}_Y \cap$ *Scanned fails some CI test in previous FindCanPC-MBC(X), which is impossible*

based on **Corollary 6.1**. *Similarly, none of* $\mathbf{PC}_Y$ *will be removed in the current FindCanPC-MBC(Y) given* **Corollary 6.1**. *Therefore, all the parents and children of* Y *will be output as connecting to* Y. ∎

Lemma 6.2 Given $G = G \cup \{(Y - X)|\forall X \in \mathbf{U}\backslash Scanned\}$, the call of *FindCanPC-MBC*(Y, G) will never output non-descendants of Y (excluding parents of Y).

Proof. *It is known from* ***Lemma 6.1*** *that* $\mathbf{PC}_Y \subseteq \{X|(Y - X) \in G\}$ *by the end of calling FindCanPC-MBC(*Y*) , which means that* $\mathbf{PC}_Y \subseteq \mathbf{PC}_Y^C$ *all alone within FindCanPC-MBC(*Y*). If there is any non-descendant* $Z \in \mathbf{ND}_Y$ *being output, then it obviously contradicts with the fact that* $I(Y, \mathbf{ND}_Y/\mathbf{Pa}_Y|\mathbf{PC}_Y)$. ∎

Theorem 6.2 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, given $G = G \cup \{(Y - X)|\forall X \in \mathbf{U}\backslash \boldsymbol{Scanned}\}$, *FindCanPC-MBC*$(Y, G)$ will always output $\mathbf{PC}_Y$ and the only possibly false positives as connected to Y in G can only be Y's descendants.

Proof. ***Lemma 6.1*** *and* ***Lemma 6.2*** *ensures that* $\boldsymbol{PC}_Y$ *will be output, and non-descendant will never be output respectively. The example that some descendants may be output can be found in the proof of* ***Theorem 3.1***. ∎

Theorem 6.3 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, $\{X|(T - X) \in \mathbb{G}_2\}$ are exactly $\mathbf{PC}_T$.

Proof. *Given* $Y \in \mathbf{PC}_T^C = \{X|(T - X) \in \mathbb{G}_1\}$, *we initialize* $G = G \cup \{(Y - X)|\forall X \in \mathbf{U}\backslash \boldsymbol{Scanned}\}$. *By calling FindCanPC-MBC(*Y, G*),* $\{X|(Y - X) \in G\}$ *is known as subset of* $\mathbf{PC}_Y \cup \mathbf{Des}_Y$. *If* $Y \in \mathbf{PC}_T$*, then it is known that* $T \in \mathbf{PC}_Y$*, and the edge* $T - Y$ *will keep left in* G. *Else if* $Y \notin \mathbf{PC}_T$*, i.e.* $Y \in \mathbf{Des}_T$*, the edge* $T - Y$ *should has be deleted since (1)* T *is then a non-descendant of* Y *and, (2) it is impossible to have this edge based on* ***Theorem 6.2***.

Because we call each FindCanPC-MBC on each $Y \in \mathbf{PC}_T^C = \{X|(T - X) \in \mathbb{G}_1\}$, *all links between* T *and* $false\ positves$ *in* $\mathbf{PC}_T^C$ *will be deleted. Therefore, what left connected to* T *in* $\mathbb{G}_2$

are only its true parents and children. In other words, we have found all links between T and $\mathbf{PC}_T$. ∎

Corollary 6.3 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, after the calling of *FindCanPC-MBC*(X, G) and *FindCanPC-MBC*(Y, G), $X - Y$ stay in G only if X and Y are truly connected.

Proof. *Please refer to the first half part of the proof on* ***Theorem 6.3*** *above.* ∎

Theorem 6.4 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, all edges existing between any $X, Y \in \mathbf{PC}_T$ in $\mathbb{G}_2$ are exactly those existing in the target $\mathbb{MBC}_T$.

Proof. *We call RecongziePC-MBC*(X, G) *on each* $X \in \boldsymbol{PC}_T^C$ *(Line 6-10), then each true* $X - Y$ *between any pair of* $X, Y \in \boldsymbol{PC}_T^C$ *should be in* $\mathbb{G}_2$. *Sicne* $\boldsymbol{PC}_T \subseteq \boldsymbol{PC}_T^C$, *then the statement gets proved.* ∎

Therefore, by the end of Step 2, we get closer to the target $\mathbb{MBC}_T$ – both $\mathbf{PC}_T$ and the links among $\mathbf{PC}_T$ are induced correctly. In addition to these, nodes connected to some $X \in \mathbf{PC}_T$ actually contain true spouses requiring for further search work, which will be discussed in Step 3. Figure 6-5 is one example of $\mathbb{G}_2$, in which the non-dotted edges and circles means they are confirmed as part of the target $\mathbb{MBC}_T$.

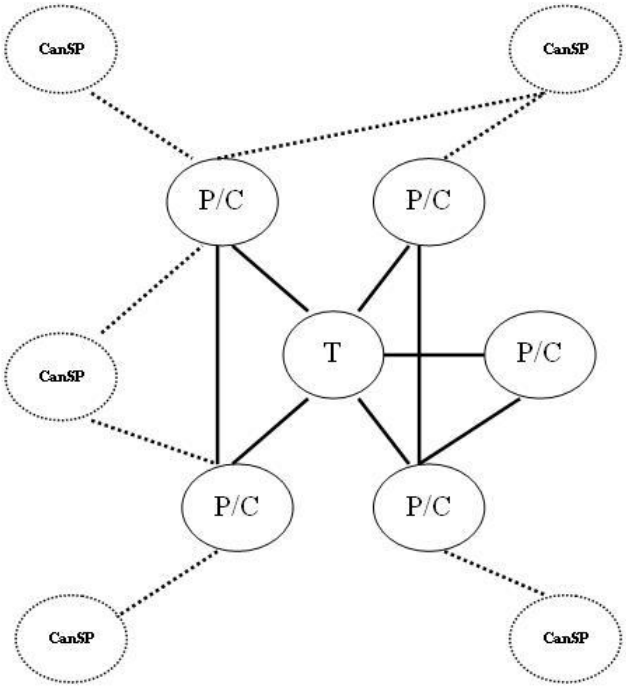


Figure 6-5: In $\mathbb{G}_2$, all connecting to T are exactly T 's parents and children, and they still cannot be distinguished further. It also contains all the possible links among PC_T. Candidate spouses Sp_T^C are found to be connected with some $X \in PC_T$. In the graph, all confirmed findings are drawn with solid lines, and non-confirmed with dotted lines.

6.4.4 Recognize Sp_T/Links among Sp_T/Links between Sp_T and PC_T

The output of Step 2, $\mathbb{G}_2$, is fed into Step 3 as input. $\mathbf{PC}_T$ can be retrieved from $\mathbb{G}_2$ easily (**Theorem 6.3**). Then, given $\forall X \in \mathbf{PC}_T$ we obtain $\mathbf{PC}_X^C$ similarly based on the connection in G. With $\forall Y \in \mathbf{PC}_X^C$, if it is recognized as true spouse (Line 15-18), *FindCanPC-MBC*(Y) is called; otherwise, it is removed from G, so as any links connecting to it. If the graph we have by the end of Step 3 is denoted as $\mathbb{G}_3$, we will prove that in addition to what true information as contained in $\mathbb{G}_2$, we have the following additional:

- All true spouses are left in $\mathbb{G}_3$, denoted as Sp_T;
- Edges between any $X, Y \in Sp_T$ are contained in $\mathbb{G}_3$;
- Edges between any $X \in Sp_T$ and $Y \in PC(T)$ are contained in $\mathbb{G}_3$.

Lemma 6.3 For $\forall X \in \mathbf{PC}_T$, $\mathbf{PC}_X^C = \{Y | (Y - X) \in G\}$ contains the spouses of T if these spouses have X as the common-child with T.

Proof. *It is known that* $\mathbf{PC}_X \subseteq \mathbf{PC}_X^C$*, so all parents of X should be contained in* $\mathbf{PC}_X^C$*, saying nothing of those having common children.* ∎

Lemma 6.4 For $\forall Y \in \mathbf{PC}_X^C$, only the true spouse of T will satisfy the condition of $I_D(T, Y | Sepset_{T,Y} \cup \{X\} \leq \varepsilon)$ (at Line 15 of IPC-MBC).

Proof. *Please refer to the proof of* **Theorem 3.4**. ∎

Theorem 6.5 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, all true spouses are correctly recognized in $\mathbb{G}_3$.

Proof. (1) *Given* $X \in \mathbf{PC}_T$*, we check each* $Y \in \mathbf{PC}_X^C$*, therefore all spouses as probably contained in* $\mathbf{PC}_X^C$ *will be correctly recognized based on* **Lemma 6.4**. *(2) The same treatment is given to each* $X \in \mathbf{PC}_T$*, hence we are able to find all spouses as contained in* $\mathbf{PC}_X^C$*. (3) Assume that there is one spouse not belonging to any* $\mathbf{PC}_X^C$*, this may happen only when the corresponding X is not contained in* $\mathbf{PC}_T$*, which obviously contradicts to* **Lemma 6.1**. ∎

Theorem 6.6 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution faithful to a DAG, all edges existing between any $X, Y \in \mathbf{Sp}_T$ in $\mathbb{G}_3$ are exactly those existing beween any $X, Y \in \mathbf{Sp}_T$ in the target $\mathbb{MBC}_T$.

Proof. *(1) All* $\mathbf{Sp}_T$ *are contained in* $\mathbb{G}_3$*, based on* **Theorem 6.5***; (2) We call FindCanPC-MBC*(X, G) *for* $\forall X \in \mathbf{Sp}_T$*, and* $G = G \cup \{(X - Y) | \forall Y \notin \mathbf{U} \backslash Scanned\}$*. So, if the edge* $X - Y$ *must be added to G before calling FindCanPC-MBC*(X, G) *assuming that X is studied earlier than Y. (3) Based on* **Corollary 6.3**, *the corresponding edge,* $X - Y$*, won't be removed from G since it is added on if it is true. (4) In contrast, all false edges between any* $X, Y \in \mathbf{Sp}_T$ *will be removed in FindCanPC-MBC*(X, G) *or FindCanPC-MBC*(Y, G)*. Therefore, what edges left between any* $X, Y \in \mathbf{Sp}_T$ *by the end of Step 3 are just the true ones.* ∎

Theorem 6.7 Under the assumptions that the independence tests are correct and that the learning data D is an independent and identically distributed sample from a probability distribution

faithful to a DAG, all edges existing between any $X \in \mathbf{PC}_T$ and any $Y \in \mathbf{Sp}_T$ in $\mathbb{G}_3$ are exactly those existing beween any $X \in \mathbf{PC}_T$ and any $Y \in \mathbf{Sp}_T$ in the target $\mathbb{MBC}_T$

Proof. *(1)It is known that all* $\mathbf{PC}_T$ *and* $\mathbf{Sp}_T$ *are contained in* $\mathbb{G}_3$ *based on previous discussion. (2) Assume that* $X \in \mathbf{PC}_T$ *and* $Y \in \mathbf{Sp}_T$*, and it is known that* $X - Y$ *is added to G when we call FindCanPC-MBC(X) at Line 8 (IPC-MBC). (3) Based on* **Corollary 6.3***, it stays in G since then if it is true; otherwise, it will be removed within FindCanPC-MBC(X) or FindCanPC-MBC(Y). If it is removed in FindCanPC-MBC(X), it will not be added back since X is marked as Scanned at Line 9, and will be ignored during preparing adjacency nodes for Y at Line 13. (4) All such true edges will be left in G since we call FindCanPC-MBC(X) for each* $X \in \mathbf{PC}_T$*. (5) All the false edges will be removed correct from G since we call FindCanPC-MBC(X) and FindCanPC-MBC(Y) for all such pairs of nodes involved.* ∎

Therefore, by the end of Step 3, (1) what left connected to T are just $\mathbf{PC}_T$, (2) All spouses are correctly recognized, (3) some children are also correctly distinguished from $\mathbf{PC}_T$ after being recognized as a common child of some $X \in \mathbf{Sp}_T$ and T; (4) all edges among $\mathbf{Sp}_T$ are recognized; and (5) all edges between $\mathbf{PC}_T$ and $\mathbf{Sp}_T$ are also recognized. In conclusion, all nodes and edges as contained in the target $\mathbb{MBC}_T$. are correctly recognized. Figure 6-6 shows the result after the additional processing, as discussed in this section, being conducted on the graph in Figure 6-5.

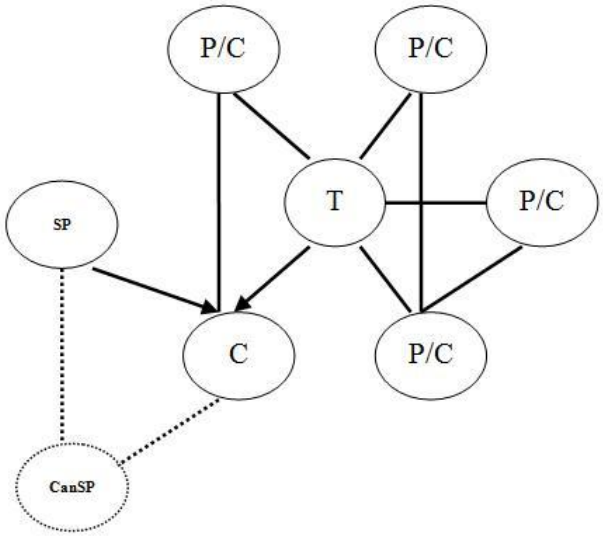


Figure 6-6: In $\mathbb{G}_3$, spouses are recognized, along with some children of T.

6.4.5 Achieve the Skeleton of $\mathbb{MBC}_T$.

All the nodes and links belonging to the target $\mathbb{MBC}_T$. are correctly recognized in G_3 by the end of last step. However, there are some by-products left in the container G_3, including nodes and links (Figure 6-6, but not all of them are presented). Removing them is trivial by judging if they belong to $\mathbf{PC}_T$ or $\mathbf{Sp}_T$. Figure 6-7 is one such example obtained from Figure 6-6, denoted as $\mathbb{G}_4$ considering it is the outcome after Step 4. $\mathbb{G}_4$ contains the skeleton (see definition below) of the target DAG with some oriented edges, but it is noted that the orientation of most links are unknown yet.

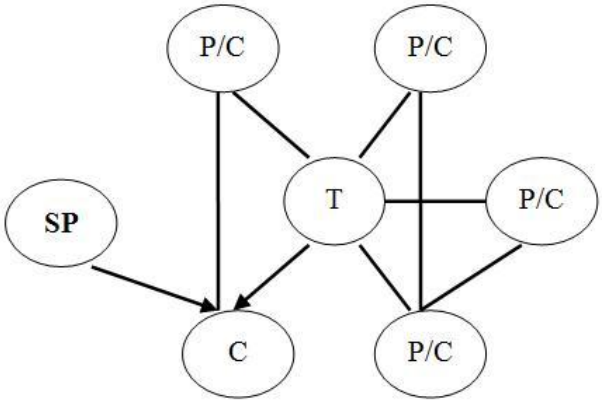


Figure 6-7: In $\mathbb{G}_4$, all the nodes and links of the target MBC are there, with some orientation determined on some links. No other nodes or links are contained.

Definition 6.2 (Skeleton). Let G be a DAG, and the undirected version of G is called the skeleton of G [52].

Definition 6.3 ($\mathbb{G}_4$). $\mathbb{G}_4$ is the skeleton of the target $\mathbb{MBC}_T$. plus orientation of some links which constructing v-structures.

6.4.6 Orientation

The orientation step will look for all triples $\{X, Y, Z\}$ such that edges $X - Z$ and $Z - Y$ are in the graph but not the edge $X - Y$. Then, if $Z \notin Sepset_{X,Y}$, we have oriented edges as $X \rightarrow Z$ and $Z \leftarrow Y$, which creates a new v-structure: $X \rightarrow Z \leftarrow Y$. After all v-structures are recognized by repeating this rule, the rest edges are oriented following two basic principles: not to create cycles and not to create new v-structure. In our implementation, we refer rules applied in Weka and [53]:

- Rule 1: $I \to J - K \ \& \ I -/- K \Rightarrow J \to K$;
- Rule 2:$I \to J \to K \ \& \ I - - K \Rightarrow I \to K$;
- Rule 3: $I \to J \leftarrow K \ \& \ I - M - K \ \& \ M - J \Rightarrow M \to$ J;
- Rule 4: $K - M - I \to J \ \& \ I - K - J \Rightarrow I \to M \leftarrow K$;
- Rule 5: if no edges are directed then take a random one (first we can find).

Lemma 6.5 For each triple $\{X, Y, Z\}$ such that edges $X - Z$ and $Z - Y$ are in the graph but not the edge $X - Y$, if $Z \notin Sepset_{X,Y}$, then we find one v-structure, i.e. $X \to Z \leftarrow Y$.

Proof. *Given a connection like $X - Z - Y$, there are three possible converges, i.e. the so-called tail-to-head $X \to Z \to Y$, head-to-head $X \leftarrow Z \to Y$ and tail-to-tail $X \to Z \leftarrow Y$. Then we need only prove that $X \to Z \to Y$ and $X \leftarrow Z \to Y$ are not possible. Assume $X \to Z \to Y$ is true, then $Z \notin Sepset_{X,Y}$; otherwise, at least there exists one path $X \to Z \to Y$ is not blocked, so X is NOT independent with Y, and $X - Y$ should exist. This is contradictory to the fact that $X - Y$ doesn't exist in graph. Similarly, we can prove that $X \leftarrow Z \to Y$ is impossible either. Therefore, $X \to Z \leftarrow Y$ must be true, and the lemma is proved.* ∎

Definition 6.4 (Markov equivalence): Two DAGs are Markov equivalent if they encode the same set of independence relations.

Theorem 6.8 Two DAGs are Markov equivalent with each other if and only if they have the same skeleton and they consist of the same of v-structure (or immoralities in the original text since they are equivalent concepts).

In the section of empirical study, we will only check the skeleton and v-structures learned when we compare them to that of the underlying true models. This simplifies the comparison work but without sacrificing the desired effect.

6.4.7 Conclusion

Our explanation on how IPC-MBC induces the target $\mathbb{MBC}_T$ is presented as step by step in this section, including how each step works, the expected outcome, and the correctness of the expected outcome. The expected result of IPC-MBC, i.e. $\mathbb{MBC}_T$, is ensured by the correctness of each step.

IPC-MBC is a direct extension of IPC-MB, but with more fine-grained control to integrating the recognition of related edges with nodes belonging to $\mathbf{MB}_T$. It completely depends on the underlying topology, and the search proceeds in a breadth-first order. By restricing the search in a local manner, IPC-MBC is expected to be more efficient than global learning algorithm like PC, and our experimental study confirms this (Section 6.6).

6.5 Complexity Analysis

The complexity of IPC-MBC is determined by the times of call on *FindCanPC-MBC*, just like *FindCanPC* to IPC-MB. The complexity of *Recognize-MBC* is same as *FindCanPC* in the worst case, that is $|\mathbf{U}| * 2^{|\mathbf{U}|-1}$. However, because $\mathrm{X} \in \mathbf{U}$ on which *FindCanPC-MBC* has been called, they may not considered in the adjacency table, instead of $\mathbf{U}\backslash\{T\}$ always in *FindCanPC* (IPC-MB), reduction on real complexity is expected depending on the underlying topology.

In IPC-MB, *FindCanPC* is only called for $\forall X \in \{T\} \cup \mathbf{PC}_T^C$; however, in IPC-MBC, *FindCanPC-MBC* is called on $\forall X \in \{T\} \cup \mathbf{PC}_T^C \cup \mathbf{Sp}_T$. Therefore, the overall time complexity of IPC-MBC is expected to be higher than IPC-MB, and the actual difference is determined by the underlying topology (since it determines the cardinality of $\mathbf{Sp}_T$). Given the example (Figure 3-5) causing the highest complexity to IPC-MB, the corresponding complexity of IPC-MBC is the same, $|\mathbf{U}|^2 * 2^{|\mathbf{U}|-1}$.

The memory complexity of IPC-MBC is similar to IPC-MB, and no more discussion is spent here.

6.6 Empirical Study

6.6.1 Experiment Design

Though IPC-MBC is proposed to induce $\mathbb{MBC}_T$, it can be regarded as an algorithm to induce $\mathbf{MB}_T$ as well. Considering that the induction of $\mathbf{MB}_T$ is the basis for inducing $\mathbb{MBC}_T$, we will firstly study the performance of IPC-MBC as a learner of Markov blanket, in term of accuracy and time efficiency. IPC-MBC will be compared with PC in our study to see how much gain in performance it has as a local search. Besides, it will be compared with IPC-MB, though it is known more complex than IPC-MB. The comparison will also give us chance to verify the implementation of IPC-MBC.

Being a MBC learner, we want to compare three approaches, including:

- PC. PC is called on **U** to induce the target BN first, then the MBC of interest can be retrieved from the BN induced;
- IPC-MB + PC. Given a target $T \in \mathbf{U}$, we run IPC-MB first to induce $MB(T)$ first, realizing dimension reduction; then, PC algorithm is employed to induce the $\mathbb{MBC}_T$ over $\mathbf{MB}_T$. In this case, only the information that $\mathbf{MB}_T \sqsubseteq \mathbf{U}$ applied, and this topology information inferred by IPC-MB is ignored;
- IPC-MBC. Given a target $T \in \mathbf{U}$, IPC-MBC is called to induce the target $\mathbb{MBC}_T$.

Considering that the output of feature selection influences the structure learning greatly, IAMB+PC is not considered due that the poor accuracy performance of IAMB. PCMB+PC is not considered as well here because PCMB has been observed with a similar performance with IPC-MB in term of accuracy, in Chapter 4.

In the experiments, we use synthetic data sampled from three networks introduced in Chapter 4 already, including Asia (Figure 4-1), Alarm (Figure 4-2), Test152 and PolyAlarm (Figure 4-3). The distribution of the size of MBC, in term of number of edges, as contained in the corresponding Bayesian network is shown in Figure 6-8. The corresponding distribution of size measured of nodes can be found in Figure 4-4. We run IPC-MB, IPC-MB + PC and IPC-MBC separately with each node in the BN as the target variable T and report the average performance over 10 rounds, including analysis of accuracy and time efficiency. PC algorithm is called for one time given a data set, and the accuracy is reported based on the $\mathbf{MB}_X$ or $\mathbb{MBC}_X$ retrieved from the Bayesian network, given each $\forall X \in \mathbf{U}$.

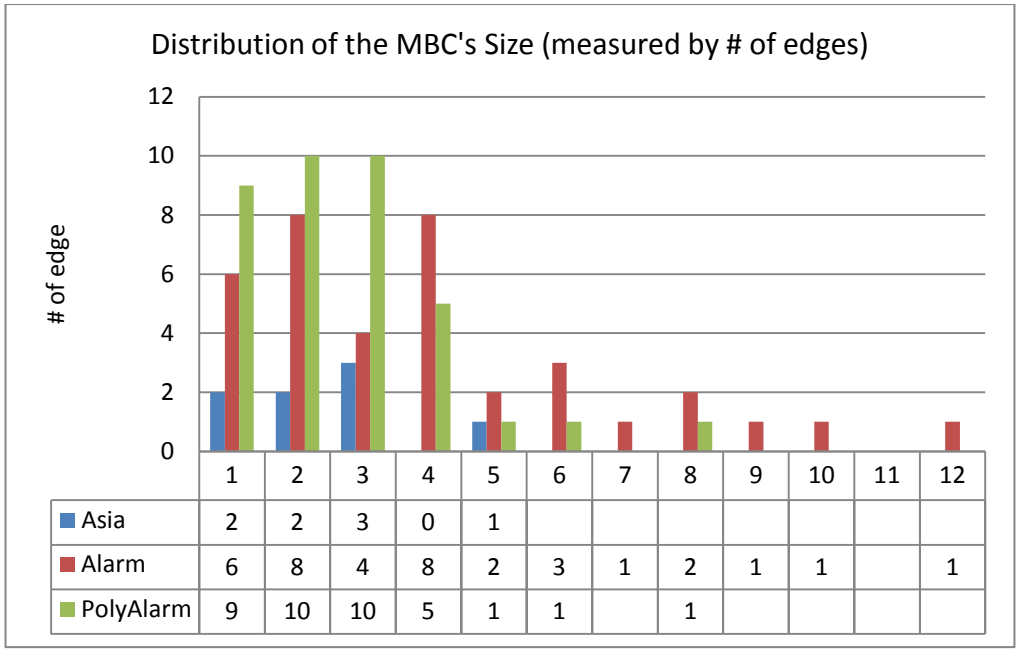


Figure 6-8: Distribution of the size of Bayesian network classifier as contained in Asia, Alarm and PolyAlarm, and the size is measured by the number of edges.

Note that in the measure of accuracy of MBC learner, we only consider the skeleton of network by ignoring the orientation of edges because (1) determining the orientation is not our research focus here, and more importantly (2) given a distribution P, there is more than one DAG encoding the same group of constraints given the equivalence network as discussed in 6.4.6.

6.6.2 IPC-MBC as Markov Blanket Learner

By applying IPC-MBC as a Markov blanket learner, we ignore the topology induced, but only check the nodes as contained in the output. In the comparison of accuracy, we still apply the measures of precision, recall and distance as we taken in Section 4.4.

Only Alarm network is used in the experiment about applying IPC-MBC as a MB learner, and the results about average accuracy and time efficiency are reported in 错误！未找到引用源。 and Table 6.2 respectively.

Table 6.1: Accuracy comparison of PC, IPC-MB and IPC-MBC over Alarm network.

Instances	Simulation Rounds	Algorithm	Precision (mean ±Std. Err)	Recall (mean ±Std. Err)	Distance (mean ±Std. Err)
500	10	PC	.77±.05	.78±.03	.37±.04

		IPC-MB	.85±02	.77±04	.32±04
		IPC-MBC	.85±02	.76±04	.33±04
1000	10	PC	.90±03	.85±03	.21±04
		IPC-MB	.94±02	.84±02	.19±03
		IPC-MBC	.94±02	.83±02	.20±03
2000	10	PC	.96±02	.90±03	.13±04
		IPC-MB	.98±02	.90±03	.11±04
		IPC-MBC	.98±02	.90±03	.11±04
3000	10	PC	.97±01	.92±02	.10±02
		IPC-MB	.99±01	.93±02	.07±03
		IPC-MBC	1.00±01	.92±02	.08±02
4000	10	PC	.97±01	.94±02	.09±03
		IPC-MB	.99±01	.95±01	.06±03
		IPC-MBC	1.00±01	.94±01	.06±01
5000	10	PC	.96±02	.94±01	.10±02
		IPC-MB	.99±01	.95±01	.05±02
		IPC-MBC	1.00±01	.94±01	.06±01

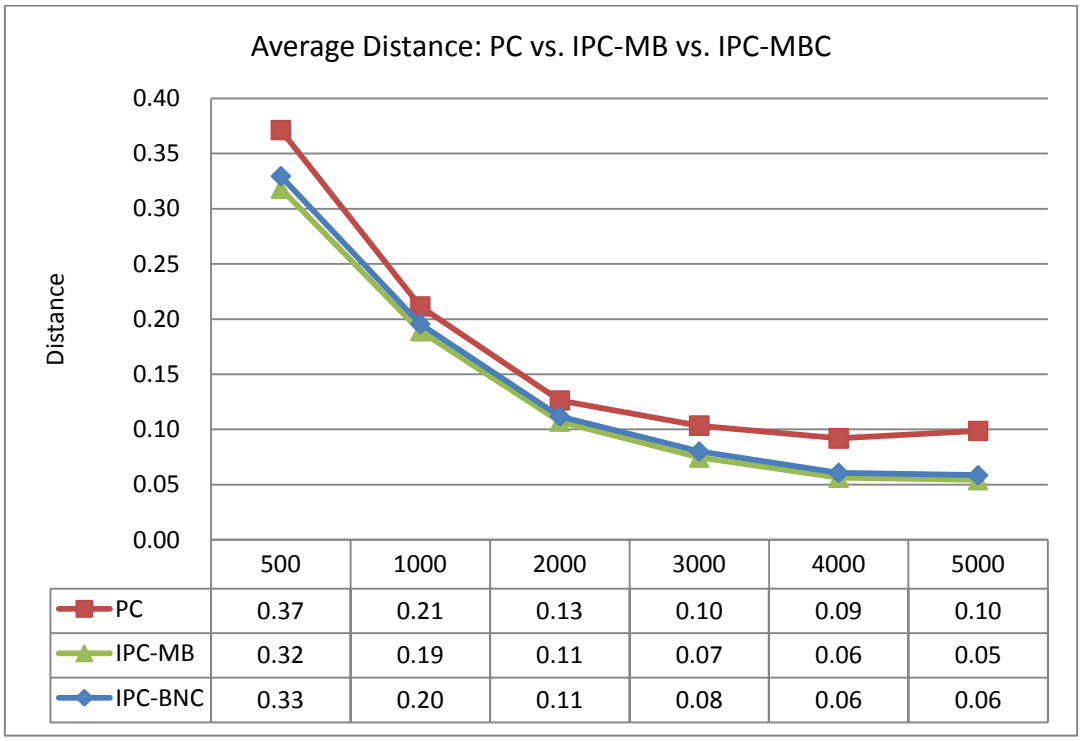
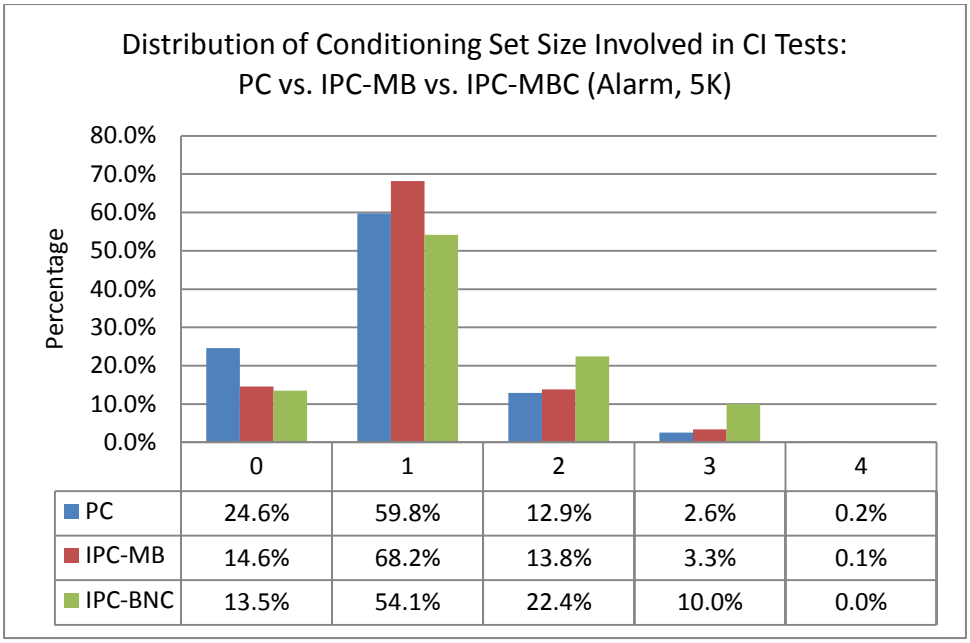


Figure 6-9: Comparison of distances given different number of instances (0.5K~5K): PC, IPC-MB and IPC-MBC (Alarm, ε= 0.05, refer to Table 6.1 for more information)

In Table 6.2, the "# CI Tests" of IPC-MB and IPC-MBC refers to the average number of CI tests we need to induce the corresponding MB given each node of the Alarm network as target. The amount for PC is the total number of CI tests required to learn the whole Alarm BN as by traditional approach. The "# Data Passes" is defined in similar way.

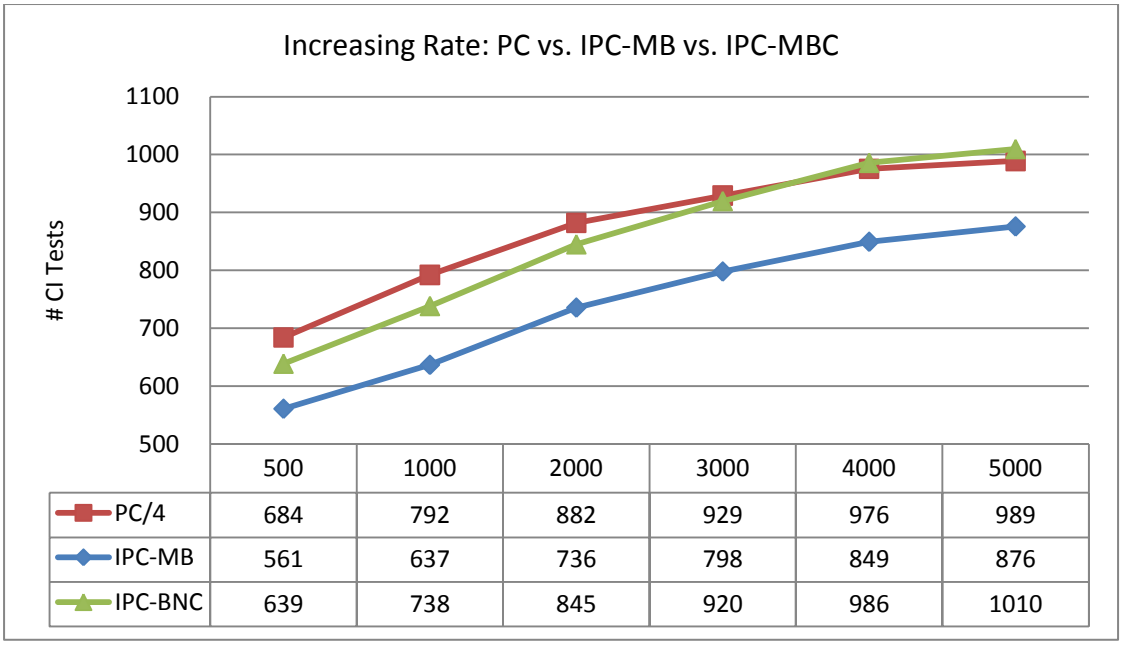
Table 6.2: Time efficiency comparison of PC, IPC-MB, IPC-MBC (Alarm, ε= 0.05).

Instances	**Simulation Rounds**	**Algorithm**	**# Data Passes (mean ±Std. Err)**	**# CI Tests (mean ±Std. Err)**
500	10	PC	220 ±16	2736 ±82
		IPC-MB	12 ±1	561 ±31
		IPC-MBC	13 ±1	639 ±32
1000	10	PC	191 ±17	3168 ±105
		IPC-MB	12 ±0	637 ±37
		IPC-MBC	15 ±0	738 ±34
2000	10	PC	188 ±12	3528 ±121
		IPC-MB	13 ±0	736 ±37
		IPC-MBC	15 ±1	845 ±39
3000	10	PC	200 ±19	3717 ±166
		IPC-MB	13 ±0	798 ±53
		IPC-MBC	16 ±0	920 ±56
4000	10	PC	211 ±18	3902 ±122
		IPC-MB	14 ±0	849 ±48
		IPC-MBC	16 ±0	986 ±64
5000	10	PC	215 ±16	3956 ±80
		IPC-MB	14 ±0	876 ±31
		IPC-MBC	16 ±0	1010 ±29



	0	1	2	3	4
PC	24.6%	59.8%	12.9%	2.6%	0.2%
IPC-MB	14.6%	68.2%	13.8%	3.3%	0.1%
IPC-BNC	13.5%	54.1%	22.4%	10.0%	0.0%

Figure 6-10: Example distribution of conditioning set size (i.e. the cardinality of conditioning set) as involved in CI tests conducted by PC, IPC-MB and IPC-MBC in experiments of Alarm (5,000 instances).



	500	1000	2000	3000	4000	5000
PC/4	684	792	882	929	976	989
IPC-MB	561	637	736	798	849	876
IPC-BNC	639	738	845	920	986	1010

Figure 6-11: Comparison of the increasing rate of CI tests as required by PC, IPC-MB and IPC-MBC given more observations (Alarm network, $\varepsilon = 0.05$). Note: For displaying and

convenient observation purpose, the corresponding number of PC algorithm is divided by 4.

Some conclusion can be made by observing the results shown in the two tables above:

1. PC, IPC-MB and IPC-MBC have nearly same accuracy performance, given different size of observations. This also reflects indirectly that they have similar data efficiency;
2. As expected, PC is the slowest one among the three, and IPC-MB is the fastest. Compared with PC, both IPC-MB and IPC-MBC realize local search, which enables them to be much more efficient than PC. For example, given 4000 observations, by average, IPC-MB and IPC-MBC require about 78% and 75% fewer number of CI tests than PC respectively; in term of data passes, IPC-MB and IPC-MBC requires about 93% and 91% fewer than that of PC respectively;
3. IPC-MBC is proposed to induce more information than IPC-MB, hence it costs more than IPC-MB. However, the additional cost is affordable as shown in our experiments with Alarm, about 10% more on both data passes and CI tests;
4. Given more observations, all three algorithms are able to conduct more searches to achieve better result, but IPC-MBC has higher increasing rate on time complexity than the other two. IPC-MB and PC has similar rate (Figure 6-11);
5. IPC-MBC is expected to achieve higher accuracy than IAMB, and faster than PCMB, based on our comparison on them with IPC-MB in Chapter 4.

Table 6.3: Accuracy comparison of PC, IPC-MB+PC and IPC-MBC over Alarm network.

Instances	**Simulation Rounds**	**Algorithm**	**Precision (mean ±Std. Err)**	**Recall (mean ±Std. Err)**	**Distance (mean ±Std. Err)**
500	10	PC	.71±.05	.74±.03	.45±.03
		IPC-MB+PC	.74±.04	.74±.04	.42±.03
		IPC-MBC	.79±.02	.73±.04	.39±.04
1000	10	PC	.88±.03	.81±.02	.26±.04
		IPC-MB+PC	.92±.02	.81±.02	.23±.06
		IPC-MBC	.93±.03	.74±.05	.29±.06
2000	10	PC	.96±.02	.87±.03	.15±.04
		IPC-	.98±.02	.87±.03	.14±.04

		MB+PC IPC-MBC	.95±.02	.77±.04	.25±.05
3000	10	PC IPC-MB+PC IPC-MBC	.97±.01 .92±.02 .99±.01	.90±.03 .92±.02 .91±.02	.12±.03 .14±.02 .10±.03
4000	10	PC IPC-MB+PC IPC-MBC	.97±.01 .92±.02 1.00±.01	.92±.01 .94±.01 .93±.02	.11±.02 .13±.03 .07±.02
5000	10	PC IPC-MB+PC IPC-MBC	.96±.02 .92±.02 1.00±.01	.93±.01 .94±.01 .94±.01	.11±.02 .12±.02 .06±.02

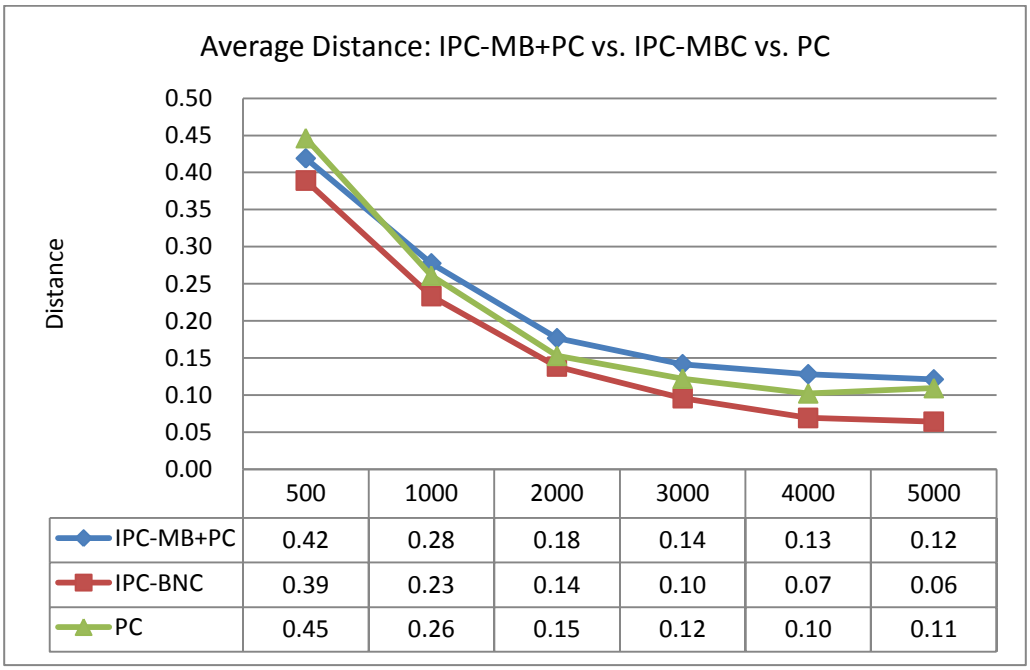


Figure 6-12: Comparison of distances given different number of instances (0.5K~5K): PC vs. IPC-MB+PC vs. IPC-MBC (Alarm, ε= 0.05, refer to Table 6.3 for more information).

Table 6.4: Accuracy comparison of PC, IPC-MB+PC and IPC-MBC over PolyAlarm network.

Instances	**Simulation Rounds**	**Algorithm**	**Precision (mean ±Std. Err)**	**Recall (mean ±Std. Err)**	**Distance (mean ±Std. Err)**

500	10	PC	.75±07	.72±05	.44±08
		IPC-MB+PC	.83±06	.74±04	.34±07
		IPC-MBC	.84±06	.74±04	.34±08
1000	10	PC	.80±04	.80±02	.34±06
		IPC-MB+PC	.90±03	.85±02	.21±04
		IPC-MBC	.91±03	.84±02	.21±04
2000	10	PC	.83±03	.83±02	.29±03
		IPC-MB+PC	.91±02	.90±01	.15±02
		IPC-MBC	.93±02	.89±02	.15±03
3000	10	PC	.83±03	.86±01	.27±03
		IPC-MB+PC	.91±04	.91±03	.15±05
		IPC-MBC	.92±03	.91±03	.14±05
4000	10	PC	.86±03	.87±03	.23±04
		IPC-MB+PC	.92±02	.92±02	.13±02
		IPC-MBC	.94±02	.91±02	.12±03
5000	10	PC	.87±03	.89±03	.20±04
		IPC-MB+PC	.92±03	.92±02	.12±03
		IPC-MBC	.94±02	.92±02	.11±02

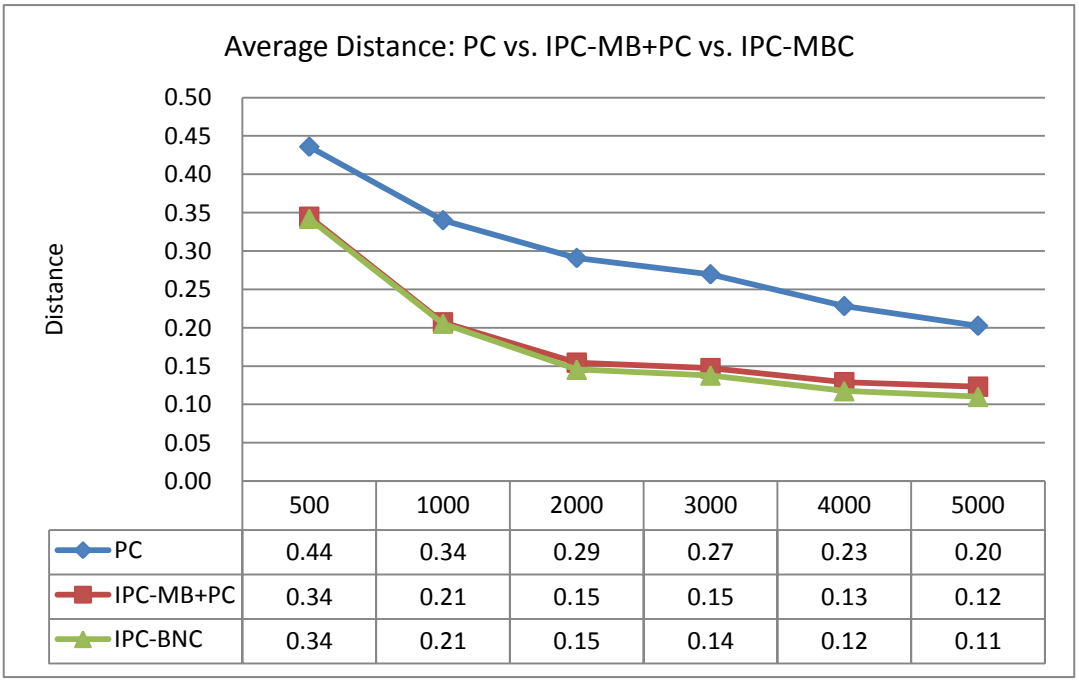
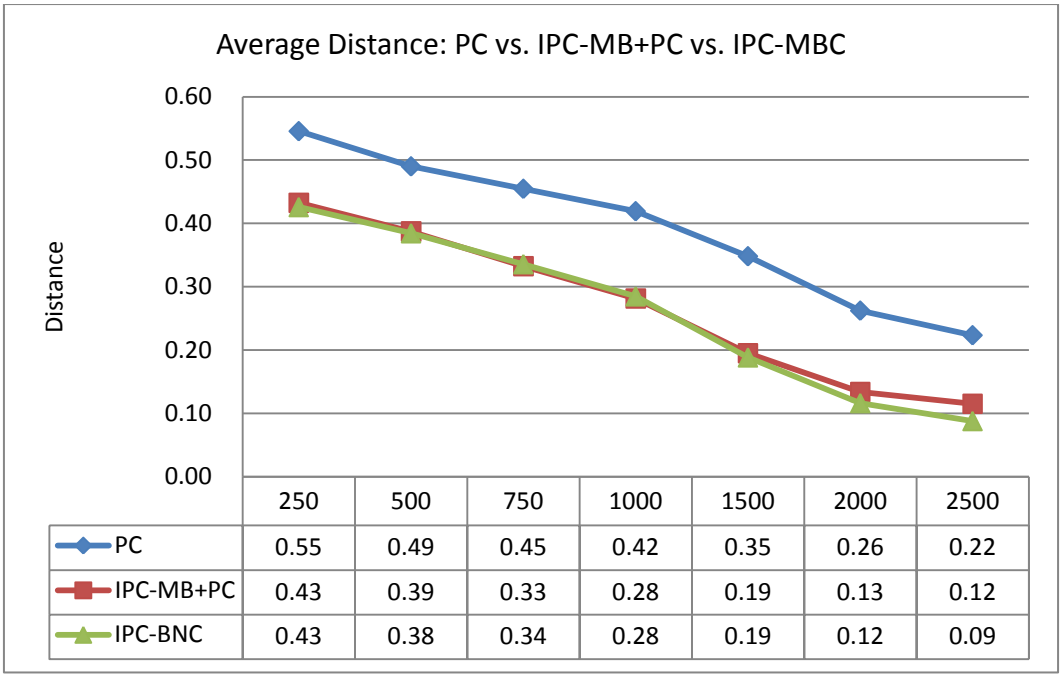


Figure 6-13: Comparison of distances given different number of instances (0.5K~5K): PC vs. IPC-MB+PC vs. IPC-MBC (PolyAlarm, ε = 0.05, refer to Table 6.4 for the complete data).

Table 6.5: Accuracy comparison of PC, IPC-MB+PC and IPC-MBC over Test152 network.

Instances	Simulation Rounds	Algorithm	Precision (mean ±Std. Err)	Recall (mean ±Std. Err)	Distance (mean ± Std. Err)
500	10	PC	.74±.03	.68±.01	.49±.01
		IPC-MB+PC	.88±.01	.69±.01	.39±.01
		IPC-MBC	.91±.01	.68±.01	.38±.01
750	10	PC	.74±.04	.73±.02	.45±.03
		IPC-MB+PC	.88±.03	.75±.02	.33±.02
		IPC-MBC	.92±.02	.72±.02	.34±.02
1000	10	PC	.74±.02	.78±.02	.42±.02
		IPC-MB+PC	.89±.02	.79±.02	.28±.02
		IPC-MBC	.93±.02	.77±.02	.28±.03
1500	10	PC	.75±.02	.87±.02	.35±.03
		IPC-MB+PC	.90±.01	.88±.03	.19±.03
		IPC-MBC	.94±.01	.86±.03	.19±.03
2000	10	PC	.78±.02	.94±.02	.26±.02
		IPC-MB+PC	.91±.01	.94±.02	.13±.02
		IPC-MBC	.95±.01	.93±.03	.12±.03
2500	10	PC	.80±.02	.97±.02	.22±.02
		IPC-MB+PC	.91±.02	.97±.01	.12±.03
		IPC-MBC	.95±.01	.96±.02	.09±.02



	250	500	750	1000	1500	2000	2500
PC	0.55	0.49	0.45	0.42	0.35	0.26	0.22
IPC-MB+PC	0.43	0.39	0.33	0.28	0.19	0.13	0.12
IPC-BNC	0.43	0.38	0.34	0.28	0.19	0.12	0.09

Figure 6-14: Comparison of distances given different number of instances (0.25K~2.5K): PC vs. IPC-MB+PC vs. IPC-MBC (Test152, ε= 0.05, refer to Table 6.5 for the complete data).

The number of data passes and CI tests as required by PC and IPC-MBC to induce the MBC here actually are same as that needed to induce the MB. IPC-MB plus PC is added here since it is not considered in the experiments about applying IPC-MBC as Markov blanket learner. Again, the measures reported on IPC-MB plus PC and IPC-MBC are the average values over each node of the target whole Bayesian network, while the values about PC are the time required to learn the whole Bayesian network. This comparison allows us to observe the difference between global and local learning.

Table 6.6: Time complexity comparison of PC, IPC-MB+PC and IPC-MBC over Asia network.

Instances	**Simulation Rounds**	**Algorithm**	**# Data Passes (mean ±Std. Err)**	**# CI Tests (mean ±Std. Err)**
100	20	PC	24 ±6	135 ±119
		IPC-MB+PC	16 ±11	129 ±166
		IPC-MBC	9 ±5	96 ±113
200	20	PC	25 ±8	149 ±136
		IPC-MB+PC	16 ±8	138 ±169
		IPC-MBC	9 ±4	110 ±136
500	20	PC	24 ±3	111 ±18

		IPC-MB+PC	15 ±3	80 ±20
		IPC-MBC	9 ±2	66 ±13
		PC	24 ±3	120 ±15
1000	10	IPC-MB+PC	16 ±3	87 ±19
		IPC-MBC	9 ±1	73 ±12
		PC	24 ±3	131 ±23
2000	10	IPC-MB+PC	16 ±4	96 ±32
		IPC-MBC	10 ±2	81 ±18
		PC	26 ±4	139 ±10
4000	10	IPC-MB+PC	15 ±2	101 ±12
		IPC-MBC	9 ±1	86 ±8
		PC	27 ±4	147 ±19
6000	10	IPC-MB+PC	17 ±2	112 ±20
		IPC-MBC	10 ±1	93 ±15
		PC	28 ±4	147 ±18
8000	10	IPC-MB+PC	17 ±2	110 ±19
		IPC-MBC	10 ±1	91 ±13
		PC	27 ±4	150 ±17
10000	10	IPC-MB+PC	14 ±2	110 ±15
		IPC-MBC	10 ±1	92 ±12
		PC	31 ±3	155 ±14
20000	10	IPC-MB+PC	19 ±2	124 ±20
		IPC-MBC	11 ±2	100 ±12

Table 6.7: Time efficiency comparison of PC, IPC-MB+PC, IPC-MBC (Alarm, ε= 0.05).

Instances	**Simulation Rounds**	**Algorithm**	**# Data Passes (mean ±Std. Err)**	**# CI Tests (mean ±Std. Err)**
		PC	220 ±16	2736 ±82
500	10	IPC-MB+PC	23 ±2	602 ±34
		IPC-MBC	13 ±1	639 ±32
		PC	191 ±17	3168 ±105
1000	10	IPC-MB+PC	24 ±1	678 ±38
		IPC-MBC	15 ±0	738 ±34
		PC	188 ±12	3528 ±121
2000	10	IPC-MB+PC	25 ±1	777 ±39
		IPC-MBC	15 ±1	845 ±39
		PC	200 ±19	3717 ±166
3000	10	IPC-MB+PC	26 ±1	844 ±55
		IPC-MBC	16 ±0	920 ±56
		PC	211 ±18	3902 ±122
4000	10	IPC-MB+PC	27 ±1	901 ±49
		IPC-MBC	16 ±0	986 ±64
5000	10	PC	215 ±16	3956 ±80
		IPC-MB+PC	27 ±1	928 ±31

	IPC-MBC	16 ±0	1010 ±29

Table 6.8: Time efficiency comparison of PC, IPC-MB+PC, IPC-MBC (PolyAlarm, ε= 0.05).

Instances	Simulation Rounds	Algorithm	# Data Passes (mean ±Std. Err)	# CI Tests (mean ±Std. Err)
500	10	PC	117 ±16	1061 ±48
		IPC-MB+PC	8 ±1	162 ±10
		IPC-MBC	13 ±1	153 ±8
1000	10	PC	140 ±26	1145 ±42
		IPC-MB+PC	15 ±0	177 ±8
		IPC-MBC	10 ±0	192 ±11
2000	10	PC	158 ±24	1223 ±35
		IPC-MB+PC	17 ±1	195 ±7
		IPC-MBC	10 ±0	212 ±8
3000	10	PC	174 ±15	1265 ±39
		IPC-MB+PC	17 ±0	209 ±8
		IPC-MBC	11 ±0	226 ±9
4000	10	PC	176 ±11	1292 ±41
		IPC-MB+PC	17 ±1	213 ±8
		IPC-MBC	11 ±0	231 ±9
5000	10	PC	181 ±2	1308 ±56
		IPC-MB+PC	18 ±1	215 ±12
		IPC-MBC	11 ±1	233 ±13

Table 6.9: Time efficiency comparison of PC, IPC-MB+PC, IPC-MBC (Test152, ε= 0.05).

Instances	Simulation Rounds	Algorithm	# Data Passes (mean ±Std. Err)	# CI Tests (mean ±Std. Err)
250	10	PC	608 ±0	17947 ±351
		IPC-MB+PC	19 ±1	796 ±29
		IPC-MBC	12 ±0	800 ±30
500	10	PC	669 ±78	19803 ±392
		IPC-MB+PC	21 ±1	946 ±29
		IPC-MBC	13 ±1	962 ±38
750	10	PC	684 ±80	21429 ±582
		IPC-MB+PC	23 ±1	1083 ±55
		IPC-MBC	15 ±1	1114 ±70
1000	10	PC	684 ±80	22732 ±426
		IPC-MB+PC	25 ±1	1179 ±34
		IPC-MBC	16 ±0	1222 ±37
1500	10	PC	714 ±73	24865 ±415
		IPC-MB+PC	27 ±1	1357 ±37
		IPC-MBC	17 ±1	1421 ±42
2000	10	PC	684 ±80	26173 ±593
		IPC-MB+PC	29 ±1	1479 ±48

		IPC-MBC	18 ±0	1556 ±57
2500	10	PC	730 ±96	27512 ±614
		IPC-MB+PC	30 ±1	1583 ±45
		IPC-MBC	18 ±0	1660 ±45

From the experimental results shown above, we notice that

- PC, IPC-MB+PC and IPC-MBC have similar accuracy performance. IPC-MBC is slightly better than the other two, and IPC-MB+PC is slightly poorer than the other two. Given more observations, all three algorithms are expected to product the perfect result;
- By applying IPC-MB to reduce the dimension first, the whole time complexity of IPC-MB + PC is much lower than applying PC directly. With the problem scale becomes larger, this saving is expected to be more obvious. For example, given Asia (20,000 instances for learning), 39% fewer of CI tests and 48% fewer of data passes are needed by IPC-MB+PC than PC; however, given larger problem like Alarm (5,000 instances for learning), the gains become as 77% and 84%;
- IPC-MBC realizes local learning as well, and the comparison with PC is discussed in 6.7. Compared with IPC-MB+PC, it has similar time complexity in term of CI tests, but obviously fewer data passes.

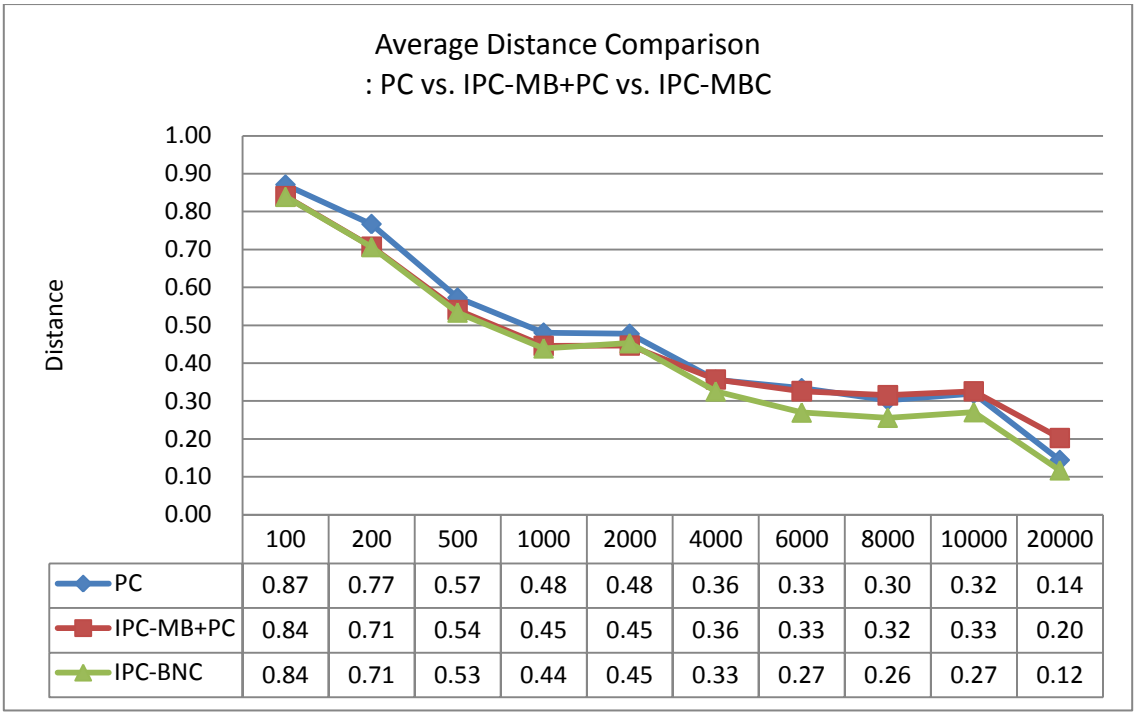
6.6.3 IPC-MBC as MBC Learner

The experiments in this section focus on the accuracy and time efficiency of IPC-MBC as MBC learner. We run IPC-MB plus PC and IPC-MBC with each node in each BN as the target variable and then, report the average precision and recall over all the nodes for each BN. **Precision** is the number of true positives in the output divided by the number of edges in the output. **Recall** is the number of true positives in the output divided by the number of true positives in the MBC. Euclidean **distance** from perfect precision and recall is defined as Equation (4.1). The significance level for the independence test is 0.05. PC algorithm is ran with one time given each data set to induce the whole network, and the precision, recall and distance are measured similarly over each MBC as retrieved from the whole BN recovered.

Table 6.10: Accuracy comparison of PC, IPC-MB+PC and IPC-MB over Asia network.

Instances	Simulation Rounds	Algorithm	Precision (mean ±Std. Err)	Recall (mean ±Std. Err)	Distance (mean ±

					Std. Err)
		PC	.40±20	.49±18	.87±13
100	20	IPC-MB+PC	.45±15	.47±17	.84±14
		IPC-MBC	.45±15	.47±17	.84±13
		PC	.49±20	.54±22	.77±13
200	20	IPC-MB+PC	.56±16	.54±12	.71±13
		IPC-MBC	.57±16	.54±11	.71±13
		PC	.65±16	.63±10	.57±16
500	20	IPC-MB+PC	.68±16	.65±11	.54±17
		IPC-MBC	.70±16	.64±11	.53±17
		PC	.70±16	.70±09	.48±16
1000	10	IPC-MB+PC	.73±15	.72±08	.45±15
		IPC-MBC	.76±17	.70±09	.44±17
		PC	.72±15	.69±05	.48±10
2000	10	IPC-MB+PC	.73±15	.73±01	.45±12
		IPC-MBC	.76±17	.69±03	.45±12
		PC	.82±05	.74±08	.36±08
4000	10	IPC-MB+PC	.81±04	.76±07	.36±06
		IPC-MBC	.87±02	.73±09	.33±10
		PC	.79±06	.81±10	.33±08
6000	10	IPC-MB+PC	.79±05	.82±09	.33±08
		IPC-MBC	.86±07	.81±10	.27±12
		PC	.82±08	.80±10	.30±11
8000	10	IPC-MB+PC	.80±06	. 82±08	.32±08
		IPC-MBC	.87±04	.80±10	.26±10
		PC	.82±09	.79±08	.32±08
10000	10	IPC-MB+PC	.80±04	.81±06	.33±03
		IPC-MBC	.87±01	.79±08	.27±07
		PC	.90±11	.93±08	.14±12
20000	10	IPC-MB+PC	.84±10	.94±08	.20±11
		IPC-MBC	.93±09	.93±08	.12±11



	100	200	500	1000	2000	4000	6000	8000	10000	20000
PC	0.87	0.77	0.57	0.48	0.48	0.36	0.33	0.30	0.32	0.14
IPC-MB+PC	0.84	0.71	0.54	0.45	0.45	0.36	0.33	0.32	0.33	0.20
IPC-BNC	0.84	0.71	0.53	0.44	0.45	0.33	0.27	0.26	0.27	0.12

Figure 6-15: Comparison of distances given different number of instances (0.1K~20K): PC vs. IPC-MB+PC vs. IPC-MBC (Asia, ε= 0.05, refer to Table 6.10 for more information)

6.7 Discussion of Different MBC Learners

Three algorithms are compared in our experiments, PC, IPC-MB+PC and IPC-MBC. Given the problem of learning $\mathbb{MBC}_T$, PC is regarded as global learning algorithm since it needs to learn the whole Bayesian network first, while the other two are viewed as local learning.

Being a typical and known algorithm for the structure learning of Bayesian network, PC is able to induce the structure efficiently. With the whole structure ready, it is trivial to get the target $\mathbb{MBC}_T$, given any $Y \in \mathbf{U}$. In our experiments, a Bayesian network is known to exist over $\mathbf{U}$, but this may not be true in real applications, especially in the exploratory stage when we are just thrown with a group of observations with feature set $\mathbf{U}$. There may exist a Bayesian network over $\mathbf{U}' \subseteq \mathbf{U}$, but not on . If $\mathbf{U}'$ is much smaller than , much resource may be wasted, though it is not avoidable. If we need only (T), and considering that $|\mathbb{MBC}_T|$ normally is even smaller than $\mathbf{U}'$, the benefit brought by local learning algorithms, in fact, will be more considerable than what we observed in the experiments here.

The combination of IPC-MB and PC is a direct application of IPC-MB as feature reduction tool, and it indeed reduces the time complexity by average, as compared with PC. For example of Alarm, IPC-MB+PC requires 77% and 84% fewer CI tests and data passes than PC's. Though PC is directly applied to the output of IPC-MB, not much work is left to PC (see Figure 6-16 for an example).

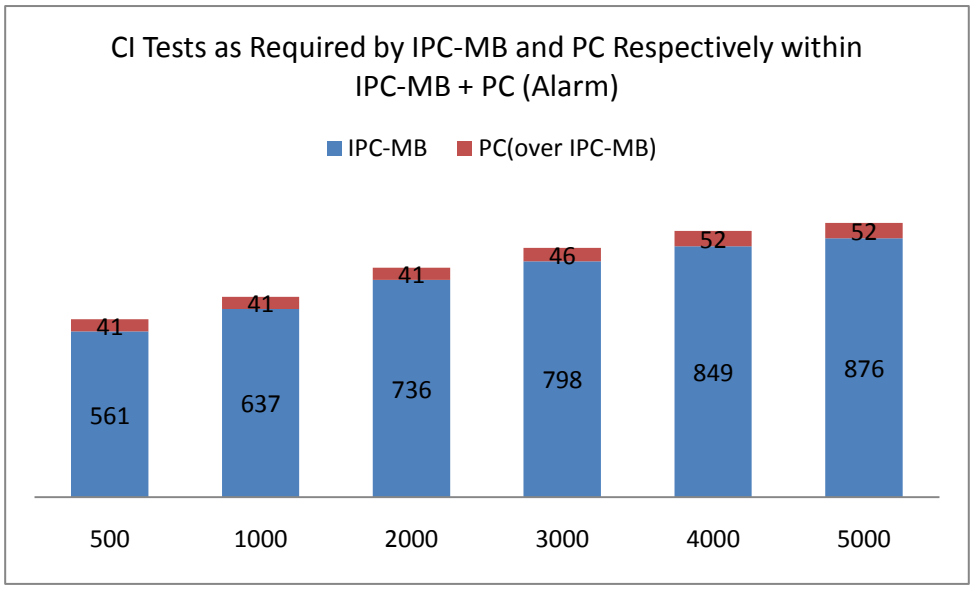


Figure 6-16: On the output of IPC-MB, the number of CI tests as required by PC to induce the connectivity is relatively small compared with that of IPC-MB.

IPC-MBC works independently, and it realizes local learning as well. Compared with IPC-MB+PC, IPC-MBC achieves a little higher accuracy performance, but requiring much fewer data passes. If we ignore the difference on data passes, whether IPC-MB+PC is more efficient than IPC-MBC is hard to say since it is influenced by the underlying topology, and we are interested to share with some in an informal way:

- Comparing with IPC-MB, the additional CI tests of IPC-MB+PC are required by PC over $\mathbf{MB}_T$, which can roughly be measured as $|\mathbf{MB}_T| * FindCanPC$; For IPC-MBC, the additional CI tests can be measured in a similar way, $|\mathbf{Sp}_T| * FindCanPC$;
- In both cases, $FindCanPC$ refers to the search within the neighborhood of some $X \in \mathbf{MB}_T$ *or* $\mathbf{Sp}_T$ in our case here. The complexity of $FindCanPC$ is determined by the cardinality of $|\mathbf{PC}_T^C|$, as well as the actual connectivity among $\mathbf{PC}_T^C$;

- The largest $|\mathbf{PC}_X^C|$ as possibly met in the PC of IPC-MB+PC is $(|\mathbf{MB}_T| - 1)$, while the smallest $|\mathbf{PC}_X^C|$ as may be met in the phase (Line 11-22, IPC-MBC) is $(|\mathbf{U}| - |\mathbf{PC}_T|)$;
- Then, if $|\mathbf{MB}_T|$ is comparable to $|\mathbf{U}|$, the work left to IPC-MB+PC will be comparable to or more than that left with IPC-MBC, when IPC-MBC will be more efficient. Asia is one such simple example, see Figure 6-7;
- Besides, if the connectivity among $\mathbf{MB}_T$, especially among $\mathbf{PC}_T$, is dense, then IPC-MBC may also be more efficient since the remaining work may be small compared with those finished.

So, it is known that the underlying topology influence the actual performance of IPC-MB+PC and IPC-MBC. However, one conclusion is confirmed, that is both of them are much more efficient than PC.

6.8 Conclusion

In this chapter, one novel algorithm called IPC-MBC is proposed to induce the Bayesian network classifier given target $T \in \mathbf{U}$, without having to learn the whole Bayesian network over $\mathbf{U}$. It is built on our work of inducing Markov blanket, IPC-MB, and hence they share similar framework, realizing the learning via a series of local search of the neighborhood of $X \in \mathbf{U}$. By carefully limiting the search in a breadth-first order, and removing as many false positives as possible in each meta-local-search, it achieves much reduction on time complexity than global learning algorithm like PC.

IPC-MB + PC is also studied in our experiments. Based on the fact that $\mathbb{MBC}_T$ is the DAG over $\mathbf{MB}_T$, applying IPC-MB first enables to reduce the search space greatly, considering that normally $\mathbf{MB}_T$ is much smaller than $\mathbf{U}$. The overall time complexity of IPC-MB+PC is observed to be much lower than PC as shown in our experiments.

In conclusion, both IPC-MB+PC and IPC-MBC are believed useful solutions to induce $\mathbb{MBC}_T$, realizing the same accuracy but requiring much less computing resource. Therefore, they are believed able to solve larger problem, or scale up better, given the same limit on CPU or memory.

Chapitre 7 CONCLUSION AND PERSPECTIVES

7.1 Conclusion on Knowledge, Work and Experience Gained

Markov Blankets are known to be the optimal feature set for a classification problem. We introduced a novel algorithm for the induction of a Markov Blanket, IPCMB, and showed that it is in general more accurate than the current state of the art algorithm, PCMB, while achieving a highly substantial performance gain. The efficiency of IPC-MB is close to the fastest, but highly inaccurate IAMB algorithm. Thus, IPC-MB offers the compelling advantage of combining speed and accuracy over the existing algorithms.

Furthermore, we showed that using the intermediate results of IPC-MB, we can derive a Markov Blanket Classifier (MBC) that is more accurate than an MBC derived by applying the classic PC algorithm to the nodes of the Markov Blanket, or by deriving the whole BN first which, in any case, is an highly inefficient solution.

7.2 Perspectives and Feature Work

7.2.1 Reduce data passes

In our implementation, due to the limit of memory and large number of contingency tables, we have to scan the data file for several times to construct necessary contingency tables to collect needed frequency information. For algorithms requiring intensive CI tests, like IPC-MB, repeating scanning the data files may be quite influential to the actual efficiency performance, especially when we have large data file. An ideal solution is to scan the data file for one time, and cache all frequencies in memory (or at least partial in memory) for later quick reference. One possibly economic choice is AD-Tree [54], and we hope to implement this to further speed up the search. If this is realized, the efficiency difference between IPC-MB and IAMB can be further reduced, making IPC-MB as more competitive a choice. Of course, we are interested to explore other effective and efficient caching solutions considering that is widely demanded in modern data mining and machine learning tasks which depend on statistical tests.

7.2.2 Work with Score-and-Search Structure Learning Algorithms

Both IPC-MBC and IPC-MB+PC are categorized into conditional test based structure learning algorithms, and untill now we haven't tried another one popular category of structure learning algorithms, i.e. the so called score-and search as we mentioned in last chapter. This family of algorithms views the structure learning of Bayesian Network as an optimization problem. They employ some measure about the consistence between the data and one graph, and add/delete/reverse edge until reach a graph with highest defined scores. However, as we discussed in Section 3.2, this approach is not suitable for identifying $\mathbf{MB}_T$ over $\mathbf{U}$, so we chose the constraint search approach in IPC-MB, which is followed by all previous works as well.

With $\mathbf{MB}_T$ ready, as produced by IPC-MB, the score-based search becomes applicable for inducing the Bayesian Network over $\mathbf{MB}_T \cup \{T\}$ (i.e. $\mathbb{MBC}_T$), just like how it works for determining the Bayesian Network over the whole problem domain $\mathbf{U}$ traditionally. In this chapter, we propose one such kind of algorithm which depends on IPC-MB to induce $\mathbf{MB}_T$ first and then induce the target $\mathbb{MBC}_T$ with score-based search. Considering that only the real effective features of the target MBC left through IPC-MB, a much smaller search space compared with the original one where all features are present, the proposed scoring-and-search learning algorithm is expected to be much more efficient than learning the whole Bayesian network with the same approach. Furthermore, compared with IPC-MB+PC, where only the information of $X \in \mathbf{MB}_T$ is referred, the additional edges and orientations information (see Section 5.7) are to be considered in this new algorithm to further narrow down the search space.

Given the output by IPC-MB, we have the knowledge about which attributes contained in the target MBC, which are parent/child nodes, and which are spouses if there are (see the following figure). Besides, from the output of IPC-MB, we have all the edges between $\mathbf{MB}_T$ and T, and some orientations as known from the induction of v-structure in IPC-MB. These known edges and orientations are fixed, which means that they won't be removed or reversed in the remaining learning by score-and-search. The overall procedure is demonstrated as in Figure 7-1.

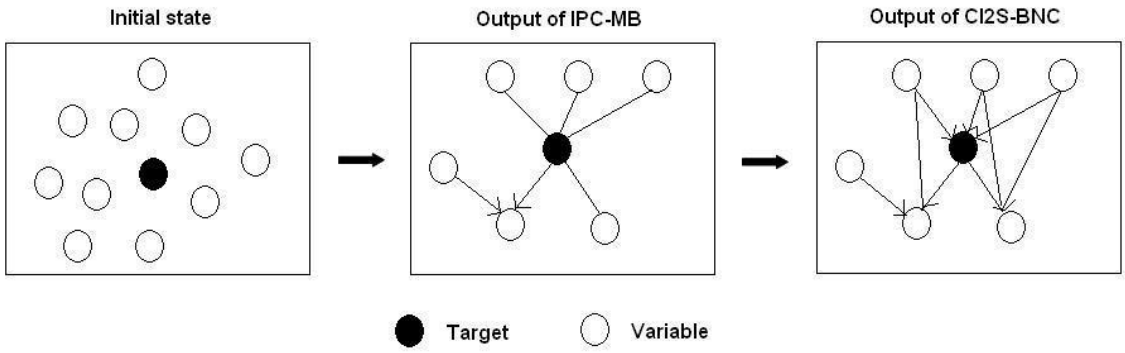


Figure 7-1: The overall procedure: start with a bag of variables, then selected with IPC-MB, and finally apply further scoring-based search to add the remaining arcs as well as to determine the orientations. v-structure determined by IPC-MB is fixed.

Two direct benefits are there to enable us to anticipate a very promising MBC learning algorithm: (1) The output of IPC-MB helps to prune the search space greatly by limiting further search only among the nodes contained in the final MBC; (2) The topology information inferred by IPC-MB further reduces the search complexity. The overall algorithm corresponding to Figure 7-1 is specified in Figure 7-3, and it can be divided into three steps:

1. Feature selection by IPC-MB;
2. Initialize the orientation of edges between P/C and T as pointing to T; the orientation of v-structure is set respectively too;
3. Apply score-and-search to reverse orientation, add edges or remove edges until no increase on score can be made. What output then is the target Bayesian network classifier.

With problem **U** and training data D, IPC-MB(T) is called to induce $\mathbf{MB}_T$ first. The correctness of IPC-MB is proved in Chapter 3, and the typical output of IPC-MB can be represented as Figure 7-2. In Figure 7-2, there are three types of information that are critical for later reference:

1. P/C nodes: They are directly connected to T, and they may be parents or children of T. Therefore, the orientation of P/C–T is unknown;
2. C nodes: They are directly connected to T, and they are known as children. Correspondinly, we have oriented arc $T \rightarrow C$;

3. SP nodes: They are not directly connected to T, but they are directly connected to C nodes. They are known as spouses of T, and we have oriented arc $S \rightarrow C$.

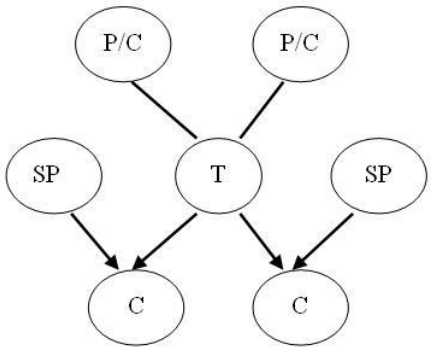


Figure 7-2: Typical output as returned by IPC-MB.

Therefore, the output of IPC-MB is very informative as compared to the initial point when we know nothing but a bag of variables **U** as given. In some cases, such kind of algorithms start with Naïve Bayes, i.e. the target variable pointing to all feature variables, which obviously includes noisy information as compared to the output of IPC-MB because normally the number of effective features is much less than the size of whole feature set.

Given the typical output of IPC-MB shown in Figure 7-2, the remaining search can be viewed as common BN structure learning, but starting with a given structure (as output by IPC-MB). To make the remaining scoring and search workable as conventional, those non-oriented arcs of the output of IPC-MB is set in advance, all pointing to the target from the P/C. Figure 7-4 is one such example derived from Figure 7-2. Compared with the target Bayesian network classifier, we need to determine additionally that:

1. If reversing the arcs $P/C \rightarrow T$ can result with higher score;
2. If there are additional edges existing between $X, Y \in \mathbf{MB}_T$, and their orientations.

In addition adding/removing/reversing edges (but without introducing a cycle), and re-calculating the score, as conducted in conventional score-and-search algorithm, two special rules must be obeyed in the search, which is specific in our solution:

- Those oriented arcs $Sp \rightarrow C \leftarrow T$ should be fixed, i.e. no deleting or reversing of orientation is applicable to them;

- Those arcs $\mathrm{P/C} \rightarrow T$ can only be reversed, but not deleted.

These two constraints further restrict the remaining search space, decreasing the problem complexity to some extent.

```
CI2S-MBC(T: target, D: dataset, ε: threshold, nIter: maximum number of iteration)
{
    //Step 1: Induce the skeleton of BNC
1.  MB_T=IPC-MB(T, D, ε);
    //Step 2: Construct the initial graph to start with
2.  G = ∅;
3.  for (each X ∈ MB_T) do
4.      if (X is kind of Pa/Ch)
5.          add X → T into G;     // It is set arbitrary to make the later scoring workable
6.      elseif(X is kind of Ch)
7.          add T → X into G;
8.      elseif(X is kind of Sp)
9.          add X → Y into G where Y ∈ MB_T and Y is known as X's child
10.     end if
11. end for
12. // Step 3: Score-based search on the basis of G
13. ScoreSearch(G, D, nIter);   // It can be hill-climbing, tabu-search etc.
14. return G;
}
```

Figure 7-3: CI2S-MBC algorithm specification

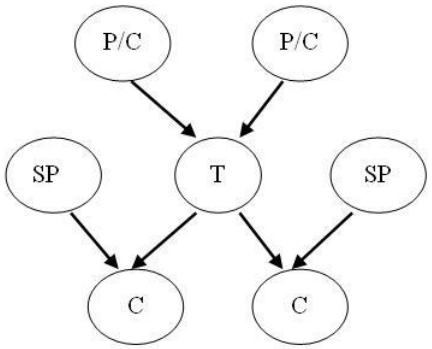


Figure 7-4: Adjust the output of IPC-MB to make the scoring work as conventional.

Implementation of CI2S-MBC is expected, including various scoring function and search strategies. A comparison with global score-and-search can be interesting, and the target algorithm is believed as another efficient algorithm for the learning of Bayesian network classifier.

7.2.3 Bayesian Network Structure Learning via Parallel Local Learning

As we discussed in 2.4, the GS [24] algorithm actually was proposed for the learning of Bayesian network via a series of local learning. The most benefit of divide-and-conquer strategy is that we are expected to solve larger scale of problems. Considering the obvious advantage of IPC-MB relative to IAMB and GS, and their similar functionality, we are interested in proposing one algorithm for the learning of Bayesian network, based on the outcome of IPC-MB.

Since the induction of Markov blanket given $X \in \mathbf{U}$ is independent one another, parallel processing is possible, which will further improve the efficiency. Besides, as we discussed in 3.7, the computing of IPC-MB can also proceed in parallel, therefore, we expect a very promising work compared with existing work.

7.2.4 Increasing the Reliability of Induction

For algorithms relying on independence information from CI tests, like IPC-MB and IPC-MBC, a major shortcoming is the impact that noise and errrors from small sample size has on the output [37]. The reliability of statistical tests significantly diminishes on small data sets. In the current version, we ignore a CI test according to the rule of Equation (1.5), and we terminate the search when there are no more reliable tests available. Taking such a conservative choice is based on empirical knowledge shared by the community as well as our own experience – conservatism is warranted by the fact that early and invalid CI tests can propagate errors by leading the search through incorrect paths.

One avenue to investigate the impact of noisy CI tests over the performance of the different algorithm is to use an Oracle in place of the CI tests : the results of each CI test would be forced to comply with the “true” distribution. Comparing the results of an oracle based simulation with that of the current method would allow to assess the impact of some of the invalid CI tests over the performance.

Bromberg and Margaritis [37] proposed a novel approach to increase the reliability of independence tests for small data sets. Their contribution is to recognize that the outcomes of independence tests are not themselves independent but are constrained by the outcomes of other tests through Pearl's well-known properties of the conditional independence relation [2]. This way, certain inconsistent test outcomes may be corrected, which will help us to avoid some errors and achieve results of higher accuracy. We are interested to incorporate their findings into our works directly or with some customizations.

7.2.5 Comparison with Other Feature Selection Algorithms

Finally, we are also interested in making a comparison with other mainstream feature selection algorithms which don't fall into this family, i.e. depending on the induction of Markov blanket. Relative effective and efficiency are two important aspects we are looking forward to a study. In term of the effectiveness, in addition to the distance measure ($\sqrt{(1-\text{precision})^2+(1-\text{recall})^2}$) taken already in our experimental studies, we will build predictors using the features chosen, and compare their relative prediction accuracy. The latter approach is useful in scenarios where we don't know the exact optimal feature subset.

BIBLIOGRAPHY

1. Koller, D. and M. Sahami. *Toward Optimal Feature Selection*. in *ICML*. 1996.
2. Pearl, J., *Probabilistic reasoning in expert systems*. 1988, San Matego: Morgan Kaufmann.
3. Chickering, D.M., D. Heckerman, and C. Meek, *Large-sample learning of Bayesian Network is NP-hard.* Journal of Machine Learning Research, 1994. **5**: p. 1287-1330.
4. Peters, D. and D.L. Parnas. *Generating a test oracle from program documentation:work in progress*. in *ACM SIGSOFT International Symposim on Software Testware Testing and Analysis*. 1994. Seattle Washington, United States: ACM.
5. Langley, P., W. Iba, and K. Thompson, *An analysis of Bayesian classifiers*, in *The Tenth National Conference of Artificial Intelligence*. 1992, AAAI Press. p. 223-228.
6. Domingos, P. and M. Pazzani. *Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier* in *International Conference on Machine Learning(ICML)*. 1996.
7. Friedman, N., D. Geiger, and M. Goldszmidt, *Bayesian Network Classifiers.* Machine Learning, 1997. **29**(2-3): p. 131-163.
8. Cheng, J. and R. Greiner. *Comparing Bayesian Network Classifiers*. in *The 15th Conference on Uncertainty in Artificial Intelligence*. 1999.
9. Fu, S.-K. and M. C.Desmarais. *Feature selection by efficient learning of Markov blanket*. in *International Conference of Data Mining and Knowledge Engieering (ICDMKE)*. 2010. London, UK: IAENG.
10. Fu, S.-K. and M.C. Desmarais. *Local Learning Algorithm for Markov Blanket Discovery*. in *Australian Conference on Artificial Intelligence*. 2007. Gold Coast, Australia: Springer.
11. Fu, S.-K. and M.C. Desmarais. *Fast Markov Blanket Discovery Algorithm Via Local Learning within Single Pass*. in *Canadian Conference on AI*. 2008. Windsor, Canada: Springer.
12. Tsamardinos, I., C.F. Aliferis, and A.R. Statnikov. *Time and sample efficient discovery of Markov blankets and direct causal relations*. in *the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2003: ACM.
13. Peña, J.M., et al., *Towards scalable and data efficient learning of Markov boundaries.* International Journal of Approximate Reasoning 2007. **45**(2).
14. Spirtes, P. and C. Glymour, *An Algorithm for Fast Recovery of Sparse Causal Graphs.* Social Science Computer Review, 1991. **9**(1): p. 62-72.
15. Fu, S.-K., M.C. Desmarais, and F. Li. *One-Pass Learning Algorithm for Fast Recovery of Bayesian Network*. in *the Twenty-First International Florida Artificial Intelligence Research Society Conference*. 2008. Coconut Grove, Florida, USA: AAAI Press.
16. Blum, A. and P. Langley, *Selection of Relevant Features and Examples in Machine Learning.* Artificial Intelligence, 1997. **97**(1-2).

17. Kohavi, R. and G.H. John, *Wrappers for Feature Subset Selection.* Artificial Intelligence, 1997. **97**(1-2).

18. Tsamardinos, I., C.F. Aliferis, and A.R. Statnikov. *Algorithms for Large Scale Markov Blanket Discovery*. in *the Sixteenth International Florida Artificial Intelligence Research Society Conference*. 2003. St. Augustine, Florida, USA: AAAI Press.

19. Guyon, I. and A. Elisseeff, *An Introduction to Variable and Feature Selection.* Journal of Machine Learning Research, 2003. **3**.

20. Breiman, L., et al., *Classification and Regression Trees*. 1 ed. 1984: Chapman & Hall.

21. Tsamardinos, I., L.E. Brown, and C.F. Aliferis, *The max-min hill-climbing Bayesian network structure learning algorithm* Machine Learning, 2006. **65**(1): p. 31-78.

22. Yaramakala, S. and D. Margaritis. *Speculative Markov Blanket Discovery for Optimal Feature Selection*. in *ICDM*. 2005.

23. Margaritis, D. and S. Thrun. *Bayesian Network Induction via Local Neighborhoods*. in *Advances in Neural Information Processing Systems* 1999. Denver, Colorado, USA: The MIT Press.

24. Margaritis, D. and S. Thrun. *Bayesian Network Induction via Local Neighborhoods* in *Neural Information Processing System (NIPS)*. 1999: MIT Press.

25. Aliferis, C.F., I. Tsamardinos, and A.R. Statnikov. *HITON: A novel Markov blanket algorithm for optimal variable selection*. in *American Medical Informatics Association Annual Symposium*. 2003.

26. Fu, S.-K. and M.C. Desmarais. *Tradeoff Analysis of Different Markov Blanket Local Learning Approaches*. in *Advances in Knowledge Discovery and Data Mining, 12th Pacific-Asia Conference (PAKDD)*. 2008. Osaka, Japan: Springer.

27. Fu, S.-K. and M. C.Desmarais. *Markov blanket based feature selection: A review of past decade*. in *International Conference of Data Mining and Knowledge Engineering (ICDMKE)*. 2010. London, UK: IAENG.

28. Spirtes, P., C. Glymour, and R. Scheines, *Causation, Prediction and Search (2nd Edition)*. 2001: The MIT Press.

29. Pearl, J. and T. Verma. *A Theory Of Inferred Causation*. in *The Second International Coference on the Principles of Knowledge Representation and Reasoning*. 1991.

30. Meek, C., *Strong completeness and faithfulness in Bayesian network*, in *the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*. 1995: Montreal, Quebec, Canada. p. 411-418.

31. Agresti, A., *Categorical Data Analysis*. 2nd ed. 2002: Wiley. 734.

32. Sokal, R.R. and F.J. Rohlf, *Bometry: The Principles and Practices of Statistics in Biological Research (3rd Edition)*. The Third Edition ed. 1994: W.H.Freeman.

33. Gross, J.L. and J. Yellen, *Handbook of Graph Theory (Discrete Mathematics and Its Application)*. 2003: CRC Press.

34. Pearl, J., *Causality: Models, Reasoning, and Inference*. 2000: Cambridge University Press.

35. Cheng, J., D. A.Bell, and W. Liu. *An algorithm for Bayesian network construction from data*. in *The 6th International Workshop on Artificial Intelligence and Statistics*. 1997.

36. Cheng, J., et al., *Learning Bayesian networks from data: An information-theory based approach.* Artificial Intelligence, 2002. **137**(1-2): p. 43-90.

37. Bromberg, F. and D. Margaritis, *Improving the Reliability of Causal Discovery from Small Data Sets Using Argumentation.* Journal of Machine Learning Research, 2009. **Special Topic on Causality**(10): p. 301-340.

38. Heckerman, D., *A Tutorial on Learning With Bayesian Networks*. 1996, Technical report of Microsoft Research.

39. Lauritzen, S. and D. Spiegelhalter, *Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems.* Journal of the Royal Statistical Society, 1988. **50**(2): p. 157-224.

40. A.Beinlich, I., et al., *The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks*, in *the 2nd European Conference on Artificial Intelligence in Medicine*. 1989. p. 247-256.

41. Abramson, B., et al., *Hailfinder: A Bayesian system for forecasting severe weather*. Probability Judgemental Forecasting, 1996. **12**(1): p. 57-71.

42. Lab, K., *Bayesian Network tools in Java (BNJ)*. 2004, Kansas State University.

43. O.Duda, R., P.E. Hart, and D.G. Stork, *Pattern Classification (2 edition)*. 2000: Wiley-Interscience. 654.

44. Langley, P. and S. Sage. *Induction of Selective Bayesian Classifiers* in *The Tenth Conference on Uncertainty in Artificial Intelligence* 1994: Morgan Kaufmann.

45. Kalisch, M. and P. B ¨uhlmann, *Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm.* Journal of Machine Learning Research, 2007. **8**: p. 613-636.

46. Cooper, G.F. and E. Herskovits, *A Bayesian method for the induction of probabilistic networks from data* Machine Learning, 1992. **9**(4): p. 309-347.

47. Heckerman, D. *A Bayesian Approach to Learning Causal Networks*. in *the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*. 1995. Montreal, Quebec, Canada: Morgan Kaufmann.

48. Robinson, R.W., *Counting unlabeled acyclic digraphs*, in *Combinatorial Mathematics V*. 1977, Springer Berlin/Hdeidelberg. p. 28-43.

49. Chickering, D.M. *A Transformational Characterization of Equivalent Bayesian*. in *The 11th Conference on Uncertainty in Artificial Intelligence (UAI)*. 1995.

50. Bouckaert, R.R., *Bayesian belief networks: from construction to inference*. 1995, University of Utrecht.

51. Chickerinig, D.M., *Learning equivalence classes of bayesian-network structures.* Journal of Machine Learning Research, 2002. **2**: p. 445-498.

52. Verma, T. and J. Pearl. *Equivalence and synthesis of causal models*. in *The Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*. 1990: Elsevier Science.

53. Verma, T. and J. Pearl. *An Algorithm for Deciding if a Set of Observed Independencies Has a Causal Explanation*. in *The 8th Conference on Uncertainty in Artificial Intelligence*. 1992: Morgan Kaufmann.

54. Moore, A. and M. Lee, *Cached sufficient statistics for efficient machine learning with large datasets.* Journal of Artificial Intelligence Research, 1988. **8**: p. 67-91.